MOTOROLA Semiconductors

MC6801

Advance Information

MOS

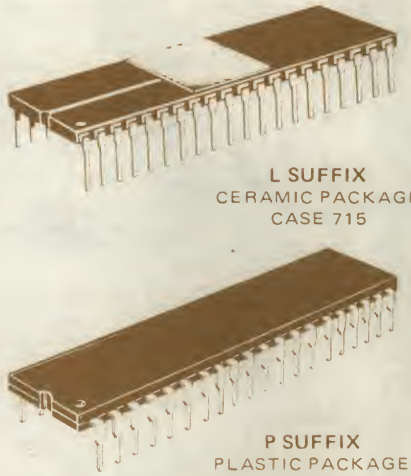
(N-CHANNEL, SILICON-GATE DEPLETION LOAD)

MICROCOMPUTER

MICROCOMPUTER UNIT (MCU)

The MC6801 MCU is an 8-bit microcomputer system which is compatible with the M6800 family of parts. The MC6801 MCU is object code compatible with the MC6800 with improved execution times of key instructions plus several new 16-bit and 8-bit instructions including an 8 X 8 unsigned multiply with 16-bit result. The MC6801 MCU can operate as a single chip microcomputer or be expanded to 65K words. The MC6801 MCU is TTL compatible and requires one +5.0 volt power supply. The MC6801 MCU has 2K bytes of ROM and 128 bytes of RAM on chip, Serial Communications Interface (S.C.I.), and parallel I/O as well as a three function 16-bit timer. Block diagram is shown in Figure 1. Features of the MC6801 include the following:

- Expanded M6800 Instruction Set
- 8 X 8 Multiply
- On-Chip Serial Communications Interface (S.C.I.)
- Object Code Compatible With The MC6800 MPU
- 16-Bit Timer
- Single Chip Or Expandable To 65K Words
- 2K Bytes Of ROM
- 128 Bytes Of RAM (64 Bytes Retainable On Power Down)
- 31 Parallel I/O Lines
- Internal Clock/Divided-By-Four
- TTL Compatible Inputs And Outputs
- Interrupt Capability
- **External Clock/Divide-By-One Mask Option (MC6801E) And EPROM Versions MC68701 And MC68701E Available Soon.**



L SUFFIX
CERAMIC PACKAGE
CASE 715

P SUFFIX
PLASTIC PACKAGE
CASE 711

FIGURE 1 - SINGLE-CHIP MICROCOMPUTER BLOCK DIAGRAM

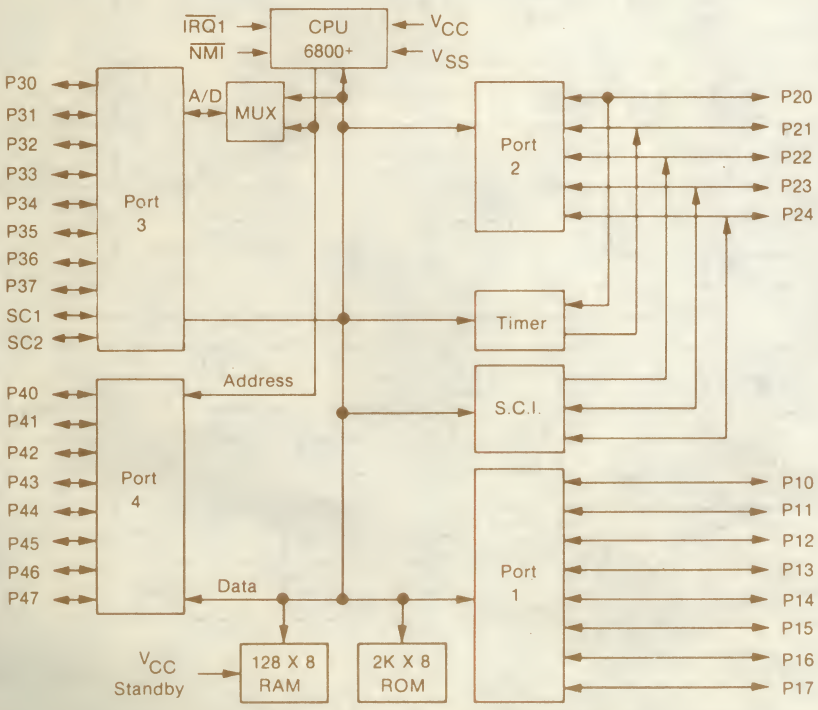
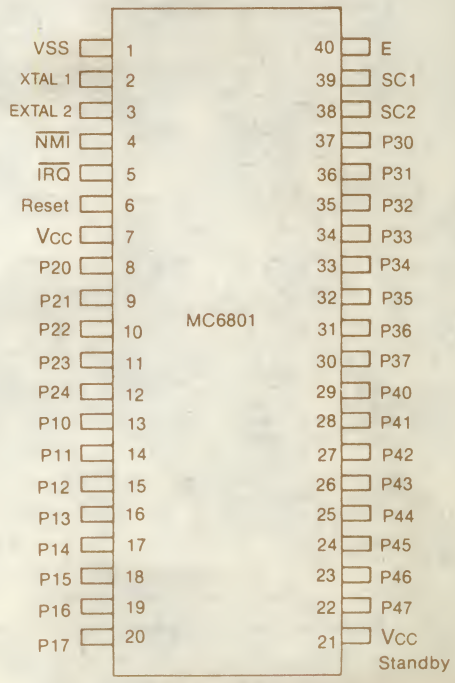


FIGURE 2 — PIN ASSIGNMENT



This is advance information and specifications are subject to change without notice.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage Reset	V_{IH}	V_{SS} + 2.0 V_{SS} + 4.0	-	V_{CC} V_{CC}	Vdc
Input Low Voltage	V_{IL}	V_{SS} - 0.3	-	V_{SS} + 0.8	Vdc
Three-State (Off State) Input Current P10-P17 (Vin = 0.4 to 2.4 Vdc) P20-P24, P30-P37	I_{TSI} I_{TSI}	- -	2.0 2.0	10 10	μAdc μAdc
Output High Voltage All Outputs Except XTAL 1 and EXTAL 2 (I_{Load} = -200 μAdc)	V_{OH}	V_{SS} + 2.4	-	-	Vdc
Output Low Voltage All Outputs Except XTAL 1 and EXTAL 2 (I Load = 1.6 mAdc)	V_{OL}	-	-	V_{SS} +0.4	Vdc
Power Dissipation	P_D	-	-	1200	mW
Capacitance (Vin = 0, T_A = 25°C, f = 1.0 MHz) P10-P17, P20-P24, P40-P47 P30-P37 Reset SC1, SC2, IRQ	C_{in}	 - - -	 - - -	 12.5 10 7.5	pF
Peripheral Data Setup Time (Figure 5)	t_{PDSU}	200	-	-	ns
Peripheral Data Hold Time (Figure 5)	t_{PDH}	0	-	-	ns
Delay Time, Enable negative transition to OS3 negative transition	t_{OSD1}	-	-	1.0	μs
Delay Time, Enable negative transition to OS3 positive transition	t_{OSD2}	-	-	1.0	μs
Delay Time, Enable negative transition to Peripheral Data Valid (Figure 6)	t_{PWD}	-	-	350	ns
Delay Time, Enable negative transition to Peripheral CMOS Data Valid (V_{CC} - 30% V_{CC}, P20-P24 (Figure 6)	t_{CMOS}	-	-	2.0	μs
Darlington Drive Current V_O = 1.5 Vdc P10-P17	I_{OH}	-1.0	-2.5	-10	mAdc
Standby Voltage (Not Operating) (Operating)	V_{SBB} V_{SB}	4.00 4.75	- -	5.25 5.25	Vdc

NOTE: The above electricals satisfy Ports 1 and 2 always, and Ports 3 and 4 in the single chip mode only.

BUS TIMING (Figure 9)

Characteristic	Symbol	Min	Typ	Max	Unit
Cycle Time	t_{CYC}	1000	-	-	ns
Address Strobe Pulse Width High	PW_{ASH}	220	-	-	ns
Address Strobe Rise Time	t_{ASR}	-	-	50	ns
Address Strobe Fall Time	t_{ASF}	-	-	50	ns
Address Strobe Delay Time	t_{ASD}	60	-	-	ns
Enable Rise Time	t_{ER}	-	-	50	ns
Enable Fall Time	t_{EF}	-	-	50	ns
Enable Pulse Width High Time	PW_{EH}	450	-	-	ns
Enable Pulse Width Low Time	PW_{EL}	450	-	-	ns
Address Strobe to Enable Delay Time	t_{ASED}	60	-	-	ns
Address Delay Time	t_{AD}	-	-	270	ns
Data Delay Write Time	t_{DDW}	-	-	225	ns
Data Set-up Time	t_{DSR}	100	-	-	ns
Hold Time } Read	t_{HR}	20	-	100	ns
Hold Time } Write	t_{HW}	20	-	-	ns
Address Delay Time for Latch	t_{ADL}	-	-	200	ns
Address Hold Time for Latch	t_{AHL}	20	-	-	ns
Pulse Width	PW_0	370	370	-	ns
Address Hold Time	t_{AH}	20	-	-	ns
Total Up Time	t_{UT}	750	-	-	ns

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	VCC	-0.3 to +7.0	Vdc
Input Voltage	Vin	-0.3 to +7.0	Vdc
Operating Temperature Range	TA	0 to 70	°C
Storage Temperature Range	Tstg	-55 to +150	°C
Thermal Resistance Plastic Package Ceramic Package	θ_{JA}	100 50	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{DD}$.

TABLE 1 — MODE AND PORT SUMMARY

MCU SIGNAL DESCRIPTION

This section gives a description of the MCU signals for the various modes. Figure 2 shows the general pin assignments for the signals. SC1 and SC2 are signals which vary with the mode that the chip is in. Table 1 gives a summary of their function.

MODE	PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	SC1	SC2
SINGLE CHIP	I/O	I/O	I/O	I/O	$\overline{IS3(I)}$	$\overline{OS3(O)}$
EXPANDED MUX	I/O	I/O	ADDRESS BUS (A0-A7) DATA BUS (D0-D7)	ADDRESS BUS* (A8-A15)	AS(O)	R/$\overline{W}$(O)
EXPANDED NON-MUX	I/O	I/O	DATA BUS (D0-D7)	ADDRESS BUS* (A0-A7)	$\overline{IOS}$(O)	R/$\overline{W}$(O)

*These lines can be substituted for I/O (Input Only) starting with the most significant address line.

I = Input
O = Output
R/$\overline{W}$ = Read/$\overline{\text{Write}}$
IS = Input Strobe
OS = Output Strobe
IOS = I/O Select
SC = Strobe Control
AS = Address Strobe

READ/WRITE TIMING FOR PORTS 3 AND 4 (Figures 3-4)

Characteristic	Symbol	Min	Typ	Max	Unit
Address Delay	t_{AD}	-	-	270	ns
Peripheral Read Access Time $t_{acc} = t_{ut} - (t_{AD} + t_{DSR})$	t_{acc}	-	-	530	ns
Data Setup Time (Read)	t_{DSR}	100	-	-	ns
Input Data Hold Time	t_{HR}	10	-	-	ns
Output Data Hold Time	t_{HW}	20	-	-	ns
Address Hold Time (Address, R/W)	t_{AH}	20	-	-	ns
Data Delay Time (Write)	t_{DDW}	-	165	225	ns
Processor Controls Processor Control Setup Time Processor Control Rise and Fall Time (Measured between 0.8V and 2.0V)	 t_{PCS} t_{PCr}, t_{PCf}	 200 -	 - -	 - 100 100	 ns ns

PORT 3 STROBE TIMING (Figures 7-8)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Strobe Delay 1	t_{DSD1}	-	-	1.0	μs
Output Strobe Delay 2	T_{OSD2}	-	-	1.0	μs
Input Strobe Pulse Width	PW_{is}	200	-	-	ns
Input Data Hold Time	t_{IH}	20	-	-	ns
Input Data Setup Time	t_{IS}	100	-	-	ns

FIGURE 3 — READ DATA FROM MEMORY OR PERIPHERALS EXPANDED NON-MULTIPLEXED

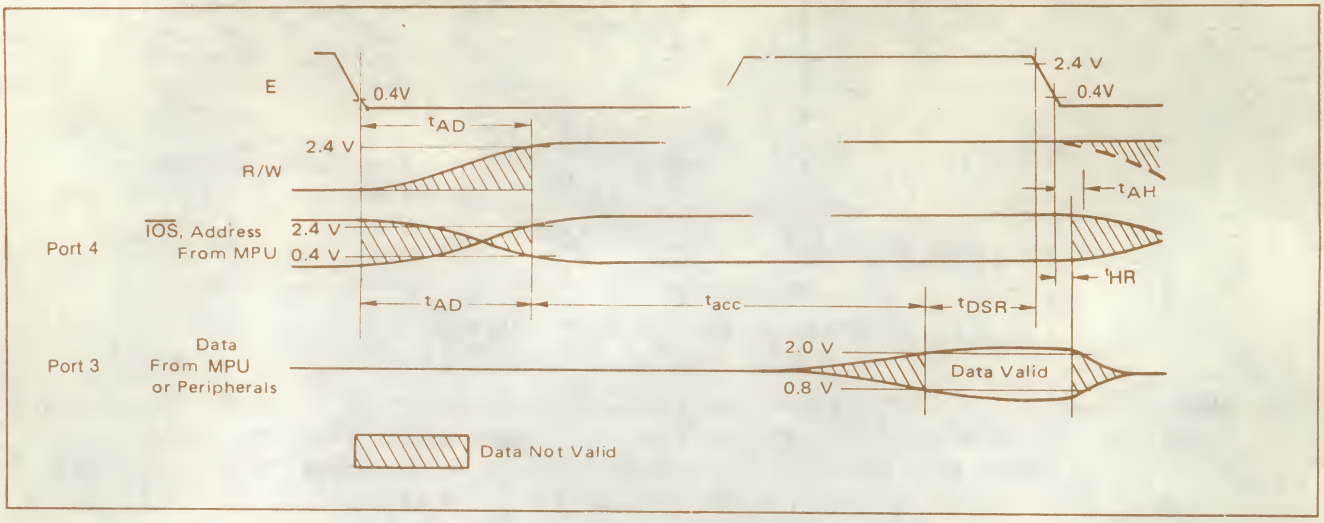
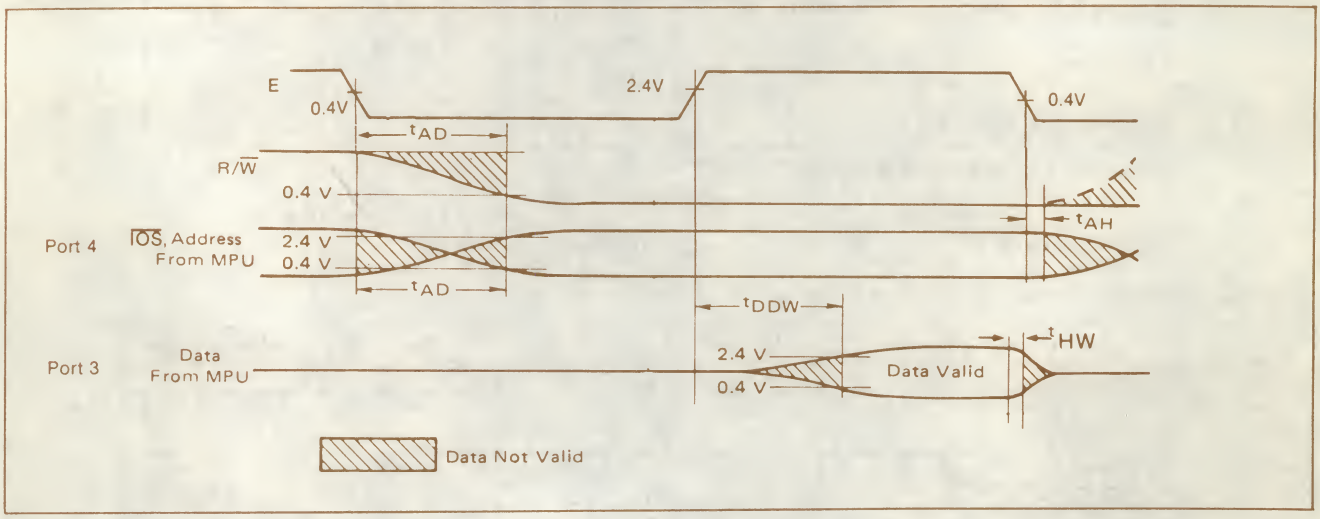


FIGURE 4 — WRITE DATA IN MEMORY OR PERIPHERALS EXPANDED NON-MULTIPLEXED



PORTS 1 AND 2 , AND PORTS 3 AND 4 IN THE SINGLE CHIP MODE

FIGURE 5 — PERIPHERAL DATA SETUP AND HOLD TIMES
(Read Mode)

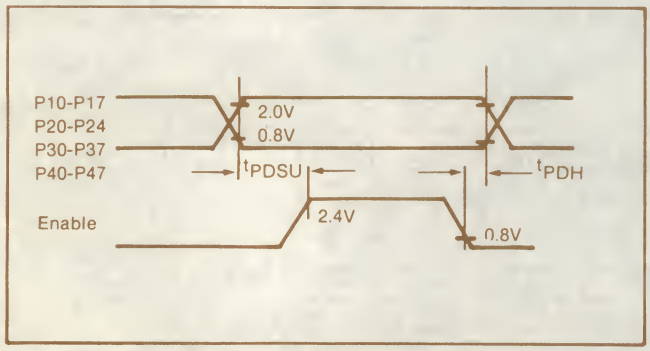


FIGURE 6 — PERIPHERAL CMOS DATA DELAY TIMES
(Write Mode)

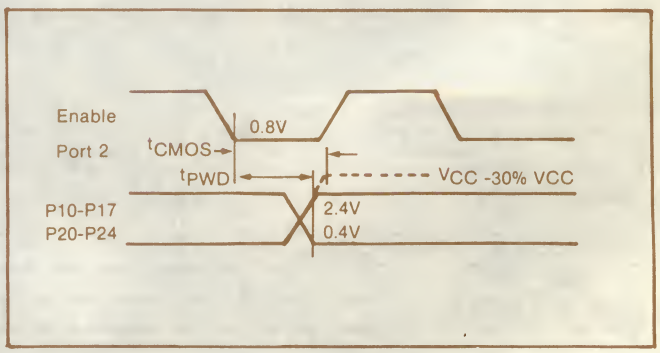


FIGURE 7 — OUTPUT STROBE TIMING — SINGLE CHIP MODE

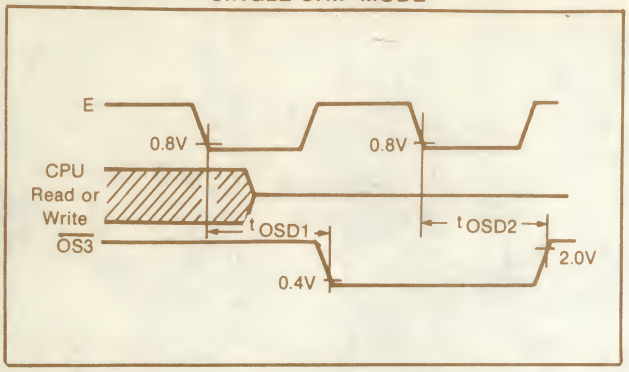


FIGURE 8 — INPUT STROBE TIMING — SINGLE CHIP MODE

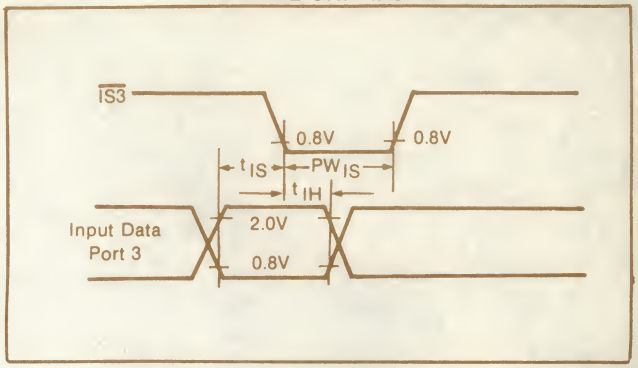


FIGURE 9 — MULTIPLEXED BUS TIMING

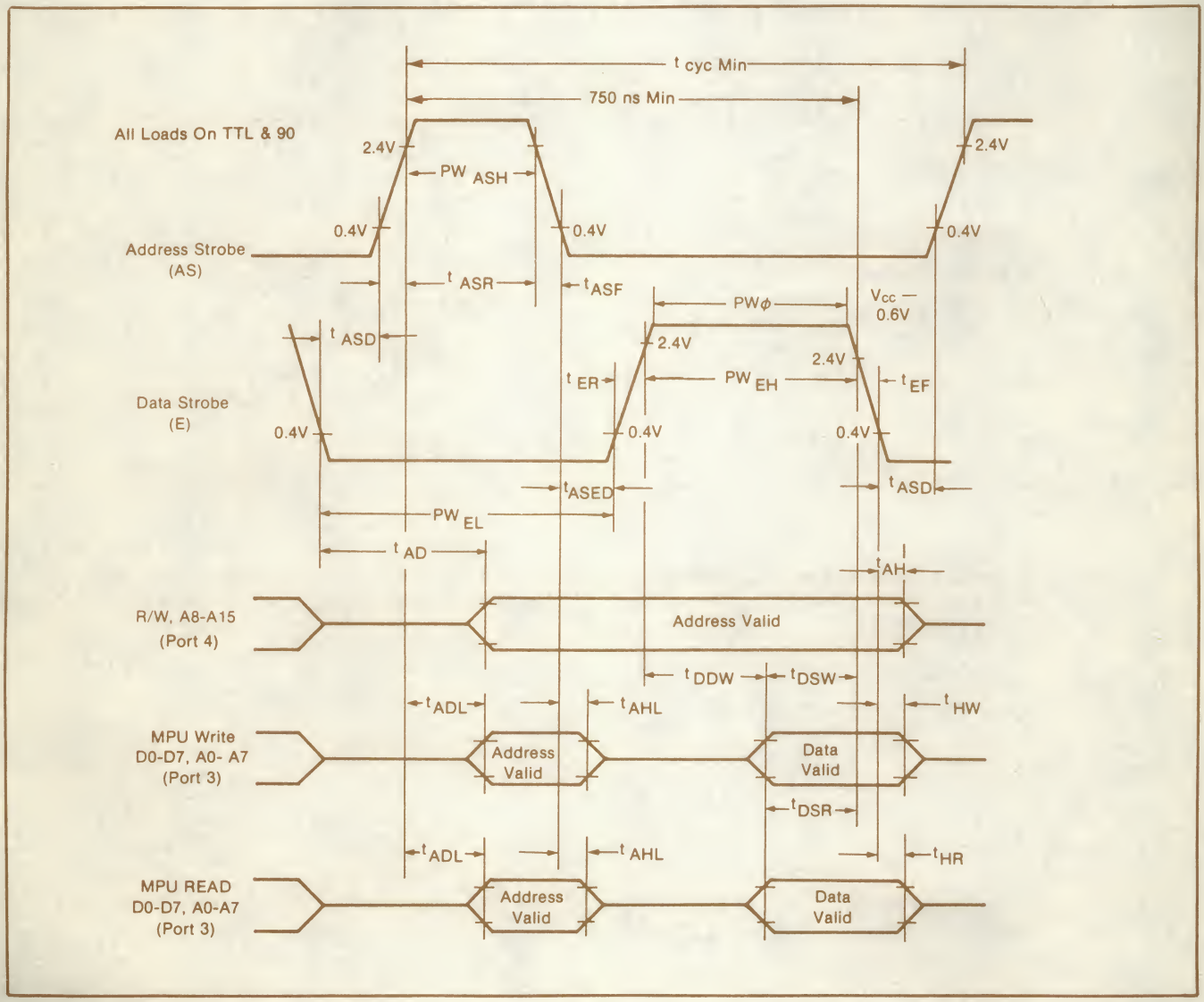


FIGURE 10—CMOS LOAD

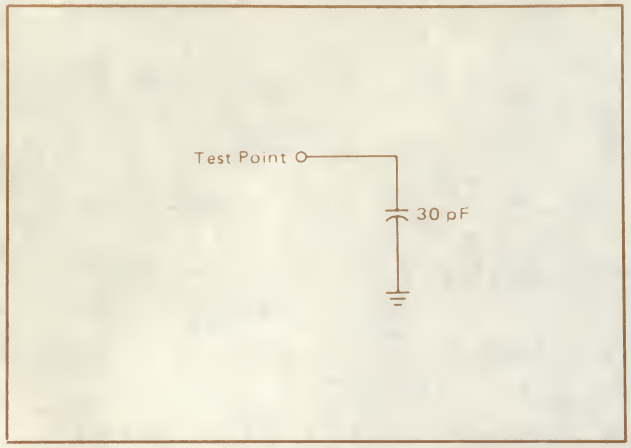
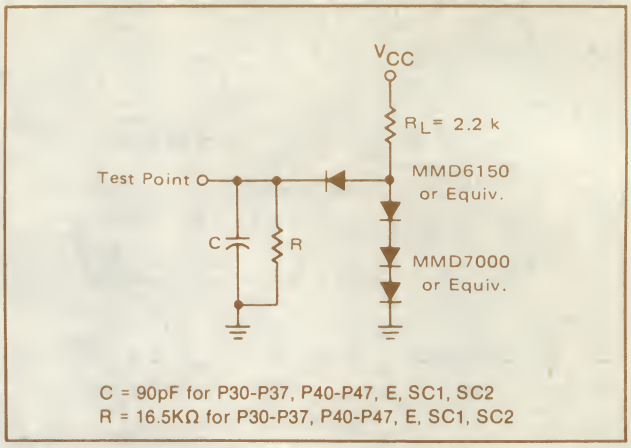


FIGURE 11 — BUS TIMING TEST LOAD AND PORTS 1, 3 AND 4 FOR SINGLE CHIP MODE



C = 90pF for P30-P37, P40-P47, E, SC1, SC2
R = 16.5KΩ for P30-P37, P40-P47, E, SC1, SC2

FIGURE 12 — TEST LOADS FOR PORT 1
Darlington Load
(P10-P17)

C = 40 R_L, R = 12k
Adjust R_L so that I_I = 3.2 mA
with V_I= 0.4V and V_{CC}=5.25 V

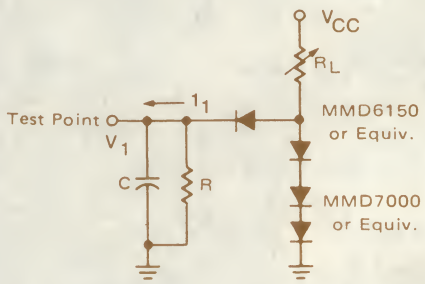


FIGURE 13 — TYPICAL DATA BUS OUTPUT DELAY versus CAPACITIVE LOADING

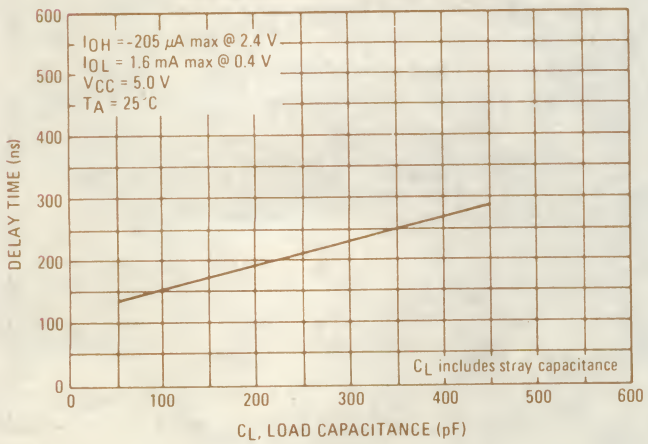
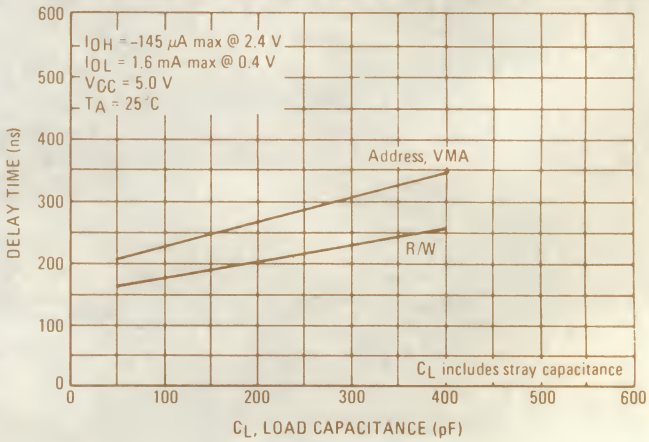


FIGURE 14 — TYPICAL READ/WRITE, VMA AND ADDRESS OUTPUT DELAY versus CAPACITIVE LOADING



SIGNAL DESCRIPTIONS

Vcc and Vss

These two pins are used to supply power and ground to the chip. The voltage supplied will be +5 volts ±5%.

XTAL 1 and EXTAL 2

These connections are for a parallel resonant fundamental crystal, AT cut. Divide by 4 circuitry is included with the internal clock, so a 4 MHz crystal may be used to run the system at 1 MHz. The divide by 4 circuitry allows for use of the inexpensive 3.56 MHz Color TV crystal for non-time critical applications. Two 27 pF capacitors are needed from the two crystal pins to ground to insure reliable operation. EXTAL2 may be driven by an external clock source at a 4 MHz rate to run at 1 MHz with a 40/60% duty cycle. It is not restricted to 4 MHz, as it will divide by 4 any frequency less than or equal to 4 MHz. XTAL1 must be grounded if an external clock is used. The following are the recommended crystal parameters:

AT = Cut Parallel Resonance Crystal
C_o = 7 pF MAX
FREQ = 4.0 MHz @ C_L = 24 pF
R_S = 50 ohms MAX.
Frequency Tolerance - ±5% to ±0.02%
The best E output "Worst Case Design" tolerance is ±0.05% (500 ppM) using A ±0.02% crystal.

Vcc Standby

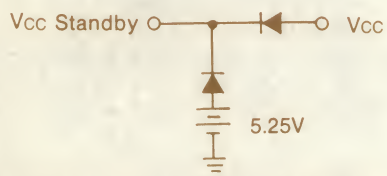
This pin will supply +5 volts ±5% to the standby RAM on the chip. The first 64 bytes of RAM will be maintained in the power down mode with 8 mA current max in the ROM version. The circuit of figure 15 can be utilized to assure that Vcc Standby does not go below VSBB during power down.

To retain information in the RAM during power down the following procedure is necessary:

1) Write "0" into the RAM enable bit, RAM E. RAM E is bit 6 of the RAM Control Register at location $0014. This disables the standby RAM, thereby protecting it at power down.

2) Keep Vcc Standby greater than VSBB.

FIGURE 15—BATTERY BACKUP FOR V_{CC} STANDBY



Reset

This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial start-up of the processor. On power up, the reset must be held low for at least 20 ms. During operation, $\overline{\text{Reset}}$, when brought low, must be held low at least 3 clock cycles.

When a high level is detected, the MPU does the following:

a) All the higher order address lines will be forced high.

b) I/O Port 2 bits, 2, 1, and 0 are latched into programmed control bits PC2, PC1 and PC0.

c) The last two (FFFE, FFFF) locations in memory will be used to load the program addressed by the program counter.

d) The interrupt mask bit is set, must be cleared before the MPU can recognize maskable interrupts.

Enable (E)

This supplies the external clock for the rest of the system when the internal oscillator is used. It is a single phase, TTL compatible clock, and will be the divide by 4 result of the crystal frequency. It will drive one TTL load and 90 pF.

Non-Maskable Interrupt ($\overline{\text{NMI}}$)

A low-going edge on this input requests that a non-maskable-interrupt sequence be generated within the processor. As with the Interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{\text{NMI}}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{\text{NMI}}$.

In response to an $\overline{\text{NMI}}$ interrupt, the Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. At the end of the sequence, a 16-bit address will be loaded that points to a vectoring address located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt service routine in memory.

A 3.3 kΩ external resistor to Vcc should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ are hardware interrupt lines that are sampled during E and will start the interrupt routine on the clock bar following the completion of an instruction.

Interrupt Request ($\overline{\text{IRQ}}$)

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further maskable interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The IRQ requires a 3.3 kΩ external resistor to Vcc which should be used for wire-OR and optimum control of interrupts. Internal Interrupts will use an internal interrupt line (IRQ2). This interrupt will operate the same as IRQ except that it will use the vector address of FFF0 and FFF7. IRQ1 will have priority over IRQ2 if both occur at the same time. The Interrupt Mask Bit in the condition mode register masks both interrupts. (See Figure 25).

The following pins are available in the Single Chip Mode, and are associated with Port 3 only.

Input Strobe ($\overline{\text{IS3}}$) (SC1)

This sets an interrupt for the processor when the IS3 Enable bit is set. As shown in Figure 8 Input Strobe Timing, IS3 will fall T_{IS} minimum after data is valid on Port 3. If IS3 Enable is set in the I/O Port Control/Status Register, an interrupt will occur. If the latch enable bit in the I/O Control Status Register is set, this strobe will latch the input data from another device when that device has indicated that it has valid data.

Output Strobe ($\overline{\text{OS3}}$) (SC2)

This signal is used by the processor to strobe an external device, indicating valid data is on the I/O pins. The timing for the Output Strobe is shown in Figure 7. I/O Port Control/Status Register is discussed in the following section.

The following pins are available in the Expanded Modes.

Read/Write (R/W) (SC2)

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read (high) or a Write (low) state. The normal standby state of this signal is Read (high). This output is capable of driving one TTL load and 90 pF.

I/O Strobe ($\overline{IOS}$) (SC1)

In the expanded non-multiplexed mode of operation, $\overline{IOS}$ internally decodes A9 through A15 as zero's and A8 as a one. This allows external access of the 256 locations from $0100 to $01FF. The timing diagrams are shown as figures 3 and 4.

Address Strobe (AS) (SC1)

In the expanded multiplexed mode of operation address strobe is output on this pin. This signal is used to latch the 8 LSB's of address which are multiplexed with data on Port 3. An 8-bit latch is utilized in conjunction with Address Strobe, as shown in figure 29, Expanded Multiplexed Mode. Address Strobe signals the latch when it is time to latch the address lines so the lines can become data bus lines during the E pulse. The timing for this signal is shown in the MC6801 Bus Timing Figure 9. This signal is also used to disable the address from the multiplexed bus allowing a deselect time, T_{ASD} before the data is enabled to the bus.

MC6801 PORTS

There are four I/O ports on the MC6801 MCU; three 8-bit ports and one 5-bit port. There are two control lines associated with one of the 8-bit ports. Each port has an associated write only Data Direction Register which allows each I/O line to be programmed to act as an input or an output.* A "1" in the corresponding Data Direction Register bit will cause that I/O line to be an output. A "0" in the corresponding Data Direction Register bit will cause that I/O line to be an input. There are four ports: Port 1, Port 2, Port 3, and Port 4. Their addresses and the addresses of their Data Direction registers are given in Table 2.

*The only exception is bit 1 of Port 2, which can either be data input or Timer output.

TABLE 2 — PORT AND DATA DIRECTION REGISTER ADDRESSES

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001
I/O Port 3	$0006	$0004
I/O Port 4	$0007	$0005

I/O Port 1

This is an 8-bit port whose individual bits may be defined as inputs or outputs by the corresponding bit in its data direction register. The 8 output buffers have three-state capability, allowing them to enter a high impedance state when the peripheral data lines are used as inputs. In order to be read properly, the voltage on the input lines must be greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0". As outputs, these lines are TTL compatible and may also be used as a source of up to 1 mA at 1.5 volts to directly drive a Darlington base. After Reset, the I/O lines are configured as inputs. In all three modes, Port 1 is always parallel I/O.

I/O Port 2

This port has five lines that may be defined as inputs or outputs by its data direction register. The 5 output buffers have three-state capability, allowing them to enter a high impedance state when used as an input. In order to be read properly, the voltage on the input lines must be greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0". As outputs, this port has no internal pullup resistors but will drive TTL inputs directly. For driving CMOS inputs, external pullup resistors are required. After Reset, the I/O lines are configured as inputs. Three pins on Port 2 (pins 10, 9 and 8 of the chip) are used to program the mode of operation during reset. The values of these pins at reset are latched into the three MSB's (bits 7, 6, and 5) of Port 2 which are read only. This is explained in the Mode Selection Section.

In all three modes, Port 2 can be configured as I/O and provides access to the Serial Communications Interface and the Timer. Bit 1 is the only pin restricted to data input or Timer output.

I/O Port 3

This is an 8-bit port that can be configured as I/O, a data bus, or an address bus multiplexed with the data bus — depending on the mode of operation hardware programmed by the user at reset. As a data bus, Port 3 is bi-directional. As an input for peripherals, it must be supplied regular TTL levels, that is, greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0".

Its TTL compatible three-state output buffers are capable of driving one TTL load and 90 pf. In the Expanded Modes, after reset, the data direction register is inhibited and data flow depends on the state of the R/W line. The input strobe (IS3) and the output strobe (OS3) used for handshaking are explained later.

In the three modes Port 3 assumes the following characteristics:

Single Chip Mode: Parallel Inputs/Outputs as programmed by its associated Data Direction Register. There are two control lines associated with this port in this mode, an input strobe and an output strobe, that can be used for handshaking. They are controlled by the I/O Port Control/Status Register explained at the end of this section.

Expanded Non-Multiplexed Mode: In this mode Port 3 becomes the data bus (D7-D0).

Expanded Multiplexed Mode: In this mode Port 3 becomes both the data bus (D7-D0) and lower bits of the address bus (A7-A0). An address strobe output is true when the address is on the port.

I/O PORT 3 CONTROL/STATUS REGISTER

	7	6	5	4	3	2	1	0
	IS3	IS3	X	OSS	LATCH	X	X	X
$000F	FLAG	ENABLE			ENABLE			

Bit 0 Not used.
Bit 1 Not used.
Bit 2 Not used.
Bit 3 **Latch Enable.** This controls the input latch for I/O Port 3. If this bit is set high the input data will be latched with the falling edge of the Input Strobe, IS3. This bit is cleared by reset, or CPU Read Port 3.

Bit 4 **(OSS) Output Strobe Select.** This bit will select if the Output Strobe should be generated by a write to I/O Port 3 or a read of I/O Port 3. When this bit is cleared the strobe is generated by a read Port 3. When this bit is set the strobe is generated by a write Port 3.

Bit 5 Not used.

Bit 6 **IS3 ENABLE.** This bit will be the interrupt caused by IS3. When set to a low level the IS3 FLAG will be set by input strobe but the interrupt will not be generated. This bit is cleared by reset.

Bit 7 **IS3 FLAG.** This is a read only status bit that is set by the falling edge of the input strobe, IS3. It is cleared by a read of the Control/Status Register followed by a read or write of I/O Port 3. Reset will clear this bit.

I/O Port 4

This is an 8-bit port that can be configured as I/O or as address lines depending on the mode of operation. In order to be read properly, the voltage on the input lines must be greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0".

As outputs, each line is TTL compatible and can drive 1 TTL load and 90 pF. After reset, the lines are configured as inputs. To use the pins as addresses, therefore, they should be programmed as outputs. In the three modes, Port 4 assumes the following characteristics:

Single Chip Mode: Parallel Inputs/Outputs as programmed by its associated Data Direction Register.

Expanded Non-Multiplexed Mode: In this mode Port 4 is configured as the lower order address lines (A7-A0) by writing one's to the data direction register. When all eight address lines are not needed, the remaining lines, starting with the most significant bit, may be used as I/O (inputs only).

Expanded Multiplexed Mode: In this mode Port 4 is configured as the high order address lines (A15-A8) by writing one's to the data direction register. When all eight address lines are not needed, the remaining lines, starting with the most significant bit, may be used as I/O (inputs only).

MODE SELECTION

The mode of operation that 6801 will operate in after Reset is determined by hardware that the user must wire on pins 10, 9, and 8 of the chip. These pins are the three LSB's (I/O 2, I/O 1, and I/O 0 respectively) of Port 2. They are latched into programmed control bits PC2, PC1, and PC0 when reset goes high. I/O Port 2 Register is shown below.

	7	6	5	4	3	2	1	0
$0003	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of external hardware that could be used in the Expanded Non-Multiplexed Mode is given in Figure 16. In the Expanded Non-Multiplexed Mode, pins 10, 9 and 8 are programmed Hi, Lo, Hi respectively as shown.

Couplers between the pins on Port 2 and the peripherals attached may be required. If the lines go to devices which require signals at power up differing from the signals needed to program the 6801's mode, couplers are necessary.

The MC14066B can be used to provide this isolation between the peripheral device and the MCU during reset. Figure 17 shows the logic diagram and truth table for the MC14066B. It is bidirectional and requires no external logic to determine the direction of the information flow. The logic shown insures that the data on the peripheral will not change before it is latched into the MCU and the MCU has started the reset sequence.

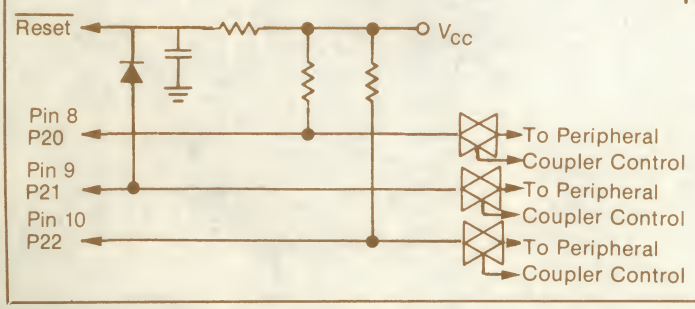
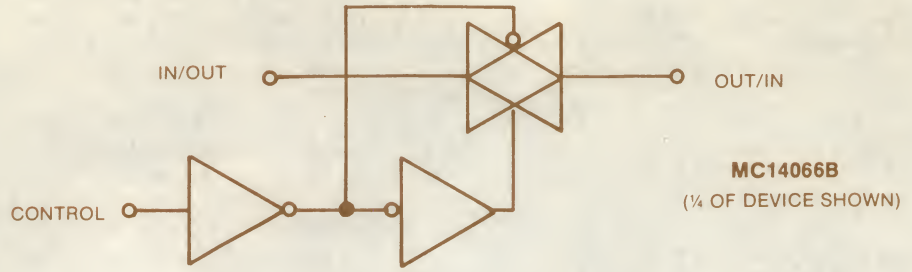


FIGURE 16 — DIODE CONFIGURATION FOR THE EXPANDED NON-MULTIPLEXED MODE

As bits 5, 6 and 7 of Port 2 are read only, the mode cannot be changed through software. The mode selections are shown in Table 3.

P20 refers to Port 2, bit 0.

FIGURE 17—MC14066B QUAD ANALOG, SWITCH/ MULTIPLEXER IN A TYPICAL MC6801 CIRCUIT



CONTROL	SWITCH
0	OFF
1	ON

V CONTROL	Vin TO Vout RESISTANCE
VSS	> 10⁹ OHMS TYP.
VDD	300 OHMS TYP.

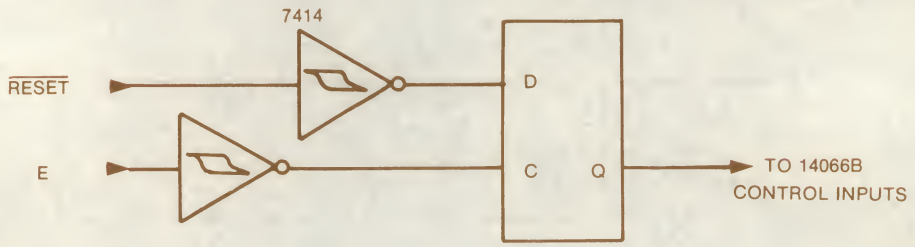
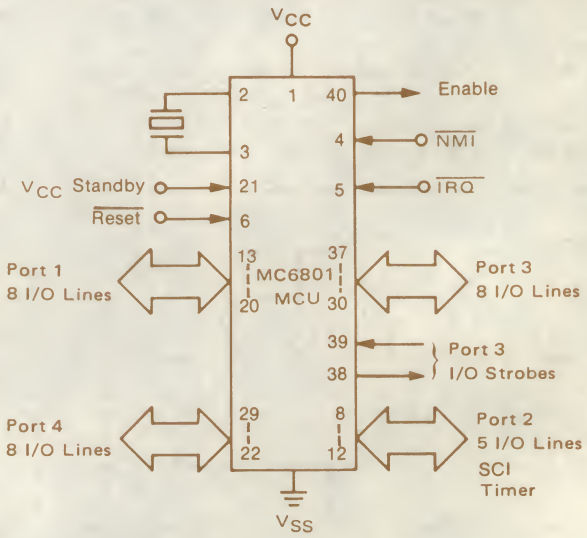


FIGURE 18 — MC6801 MCU SINGLE-CHIP MODE



MC6801 BASIC MODES

The MC6801 is capable of operating in three basic modes; (1) Single Chip Mode, (2) Expanded Multiplexed Mode (compatible with M6800 peripheral family) (3) Expanded Non-Multiplexed Mode.

SINGLE CHIP MODE

In the Single Chip Mode the Ports are configured for I/O.

This is shown in Figure 18 the single Chip Mode. In this mode, Port 3 will have two associated control lines, an input strobe and an output strobe for handshaking data.

EXPANDED NON-MULTIPLEXED MODE

In this mode the MC6801 will directly address M6800 peripherals with no external logic. In this mode Port 3 becomes the data bus, Port 4 becomes the A7-A0 address bus or partial address and I/O (inputs only), Port 2 can be parallel I/O, serial I/O, Timer, or any combination thereof. Port 1 is parallel I/O only. In this mode the MC6801 is expandable to 256 locations. The eight address lines associated with Port 4 may be substituted for I/O (inputs only) if a fewer number of address lines will satisfy the application. (See Figure 19).

FIGURE 19 — MC6801 MCU EXPANDED NON-MULTIPLEXED MODE

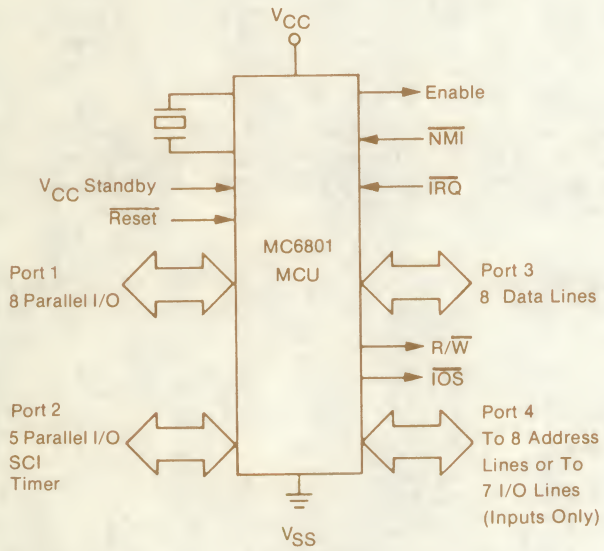
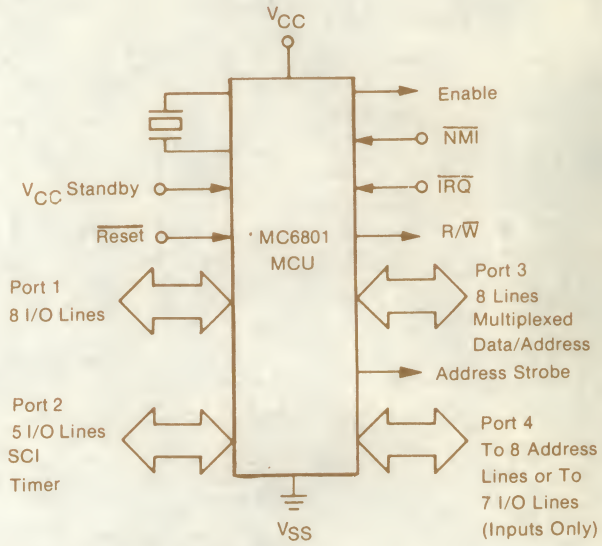


FIGURE 20 — MC6801 MCU EXPANDED MULTIPLEXED MODE



EXPANDED MULTIPLEXED MODE

In this mode Port 4 becomes higher order address lines with an alternative of substituting some of the address lines for I/O (inputs only). Port 3 is the data bus multiplexed with the lower order address lines differentiated by an output called Address Strobe. Port 2 is 5 lines of Parallel I/O, SCI, Timer, or any combination thereof. Port 1 is 8 Parallel I/O lines. In this mode it is expandable to 65K words. (See Figure 20).

TABLE 3 — MODE SELECTS

MODE		PROGRAM CONTROL			ROM	RAM	INTERRUPT VECTORS	BUS
7	SINGLE CHIP	Hi	Hi	Hi	I	I	I	I
6	EXPANDED MULTIPLEXED	Hi	Hi	Lo	I	I	I	Ep/M
5	EXPANDED NON-MULTIPLEXED	Hi	Lo	Hi	I	I	I	Ep
4	SINGLE CHIP TEST	Hi	Lo	Lo	I(2)	I(1)	I	I
3	64K ADDRESS I/O	Lo	Hi	Hi	E	E	E	Ep/M
2	PORTS 3 & 4 EXTERNAL	Lo	Hi	Lo	E	I	E	Ep/M
1		Lo	Lo	Hi	I	I	E	Ep/M
0	TEST-DATA OUTPUTTED FROM ROM & RAM TO I/O PORT 3	Lo	Lo	Lo	I	I	I*	Ep/M

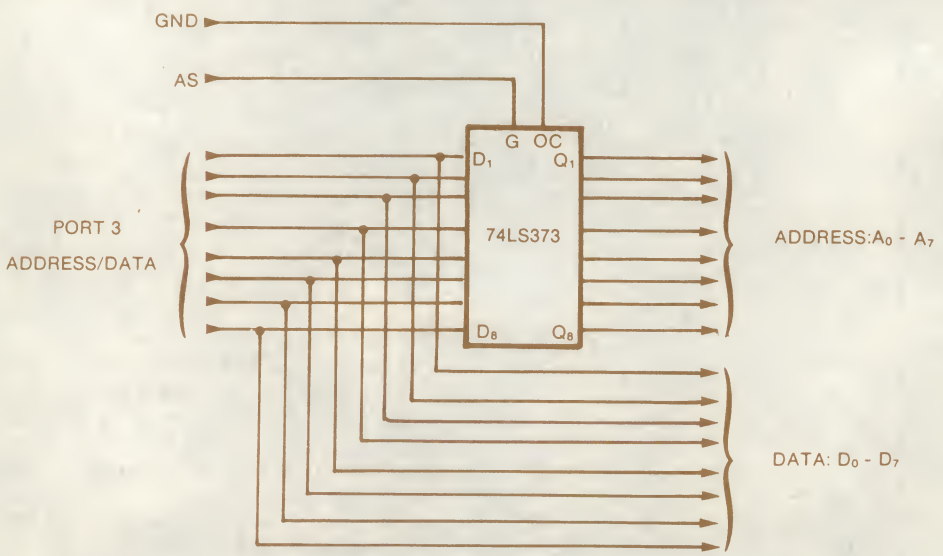
E — EXTERNAL all vectors are external
I — INTERNAL
Ep — EXPANDED
M — MULTIPLEXED

* First two addresses read from external after reset
(1) Address for RAM XX80-XXFF
(2) ROM disabled

Lower order Address Bus Latches

Since the data bus is multiplexed with the lower order address bus in Port 3, latches are required to latch those address bits. The SN74LS373 Transparent octal D-type latch can be used wtih the MC6801 to latch the least significant address byte. Figure 21 shows how to connect the latch to the MC6801. The output control to the LS373 may be connected to ground.

FIGURE 21 — LATCH CONNECTION



FUNCTION TABLE

OUTPUT CONTROL	ENABLE G	ENABLE D	OUTPUT Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

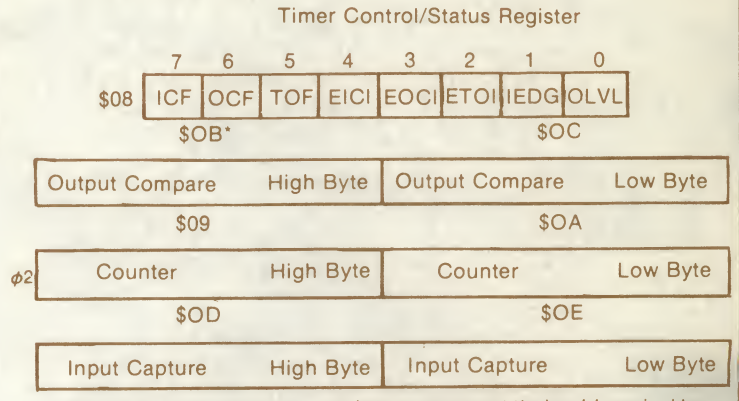
PROGRAMMABLE TIMER

The MC6801 contains an on-chip 16-bit programmable timer which may be used to perform measurements on an input waveform while independently generating an output waveform. Pulse widths for both input and output signals may vary from a few microseconds to many seconds. The timer hardware consists of

- an 8-bit control and status register,
- a 16-bit free running counter,
- a 16-bit output compare register, and
- a 16-bit input capture register

A block diagram of the timer registers is shown in Figure 22.

FIGURE 22 — BLOCK DIAGRAM OF TIMER REGISTERS



* The characters above the registers represent their address in Hex.

Free Running Counter ($0009:000A)

The key element in the programmable timer is a 16-bit free running counter which is driven to increasing values by the MPU ϕ. The counter value may be read by the MPU software at any time. The counter is cleared to zero on RESET and may be considered a read-only register with one exception. Any MPU write to the counter's address ($09) will always result in a preset value of $FFF8 being loaded into the counter regardless of the value involved in the write. This preset feature is intended for testing operation of the part, but may be of value in some applications.

Output Compare Register ($000B:000C)

The Output Compare Register is a 16-bit read/write register which is used to control an output waveform. The contents of this register are constantly compared with the current value of the free running counter. When a match is found, a flag is set (OCF) in the Timer Control and Status Register (TCSR) and the current value of the Output Level bit (OLVL) in the TCSR is clocked to the output level register. Providing the Data Direction Register for Port 2, Bit 1 contains a "1" (output), the output level register value will appear on the pin for Port 2 Bit 1. The values in the Output Compare Register and Output level bit may then be changed to control the output level on the next compare value. The Output Compare Register is set to $FFFF during RESET. The Compare function is inhibited for one cycle following a write to the high byte of the Output Compare Register to insure a valid 16-bit value is in the register before a compare is made.

Input Capture Register ($000D:000E)

The Input Capture Register is a 16-bit read-only register used to store the current value of the free running counter when the proper transition of an external input signal occurs. The input transition change required to trigger the counter transfer is controlled by the Input Edge bit (IEDG) in the TCSR. The Data Direction Register bit for Port 2 Bit 0, should* be clear (zero) in order to gate in the external input signal to the edge detect unit in the timer.

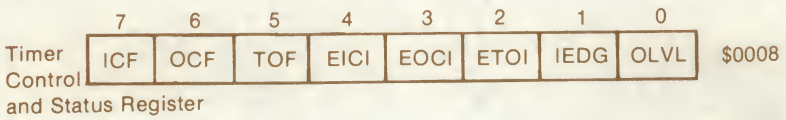
*With Port 2 Bit 0 configured as an output and set to "1", the external input will still be seen by the edge detect unit.

Timer Control and Status Register (TCSR) ($0008)

The Timer Control and Status Register consists of an 8-bit register of which all 8 bits are readable but only the low order 5 bits may be written. The upper three bits contain read-only timer status information and indicate that:

- a proper transition has taken place on the input pin with a subsequent transfer of the current counter value to the input capture register,
- a match has been found between the value in the free running counter and the output compare register, and
- when $0000 is in the free running counter.

Each of the flags may be enabled onto the MC6801 internal bus (IRQ2) with an individual Enable bit in the TCSR. If the I-bit in the MC6801 Condition Code register has been cleared, a prioriy vectored interrupt will occur corresponding to the flag bit(s) set. A description for each bit follows:



Bit 0 **OLVL** Output Level — This value is clocked to the output level register on an output compare. If the DDR for Port 2 bit 1 is set, the value will appear on the output pin.

Bit 1 **IEDG** Input Edge — This bit controls which transition of an input will trigger a transfer of the counter to the input capture register. The DDR for Port 2 Bit 0 must be clear for this function to operate.
IEDG = 0 Transfer takes place on a negative (high-to-low transition).
IEDG = 1 Transfer takes place on a positive edge (low-to-high transition).

Bit 2 **ETOI** Enable Timer Overflow Interrupt — When **set**, this bit enables IRQ2 to occur on the internal bus for a TOF interrupt; when **clear** the interrupt is inhibited.

Bit 3 **EOCI** Enable Output Compare Interrupt — When **set**, this bit enables IRQ2 to appear on the internal bus for an input capture interrupt; when **clear** the interrupt is inhibited.

Bit 4 **EICI** Enable Input Capture Interrupt — When **set**, this bit enables IRQ2 to occur on the internal bus for an input capture interrupt; when **clear** the interrupt is inhibited.

Bit 5 **TOF** Timer Overflow Flag — This read-only bit is **set** when the counter contains $0000. It is **cleared** by a read of the TCSR (with TOF set) followed by an MPU read of the Counter ($09).

Bit 6 **OCF** Output Compare Flag — This read-only bit is **set** when a match is found between the output compare register and the free running counter. It is **cleared** by a read of the TCSR (with OCF set) followed by an MPU write to the output compare register ($0B or $0C).

Bit 7 **ICF** Input Capture Flag — This read-only status bit is **set** by a proper transition on the input to the edge detect unit; it is **cleared** by a read of the TCSR (with ICF set) followed by an MPU read of the Input Capture Register ($0D).

SERIAL COMMUNICATIONS INTERFACE

The MC6801 contains a full-duplex asynchronous serial communications interface (SCI) on board. Two serial data formats (standard mark/space (NRZ) or Bi-phase) are provided at several different data rates. The controller comprises a transmitter and a receiver which operate independently or each other but in the same data format and at the same data rate. Both transmitter and receiver communicate with the MPU via the data bus and with the outside world via pins 2, 3, and 4 of Port 2. The hardware, software, and registers are explained in the following paragraphs.

Wake-Up Feature

In a typical multi-processor application, the software protocol will usually contain a destination address in the initial byte(s) of the message. In order to permit non-selected MPU's to ignore the remainder of the message, a wake-up feature is included whereby all further interrupt processing may be optionally inhibited until the beginning of the next message. When the next message appears, the hardware re-enables (or "wakes-up") the for the next message. The "wake-up" is automatically triggered by a string of ten consecutive 1's which indicates an idle transmit line. The software protocol must provide for the short idle period between any two consecutive messages.

Programmable Options

The following features of the MC6801 serial I/O section are programmable:

- format — standard mark/space (NRZ) or Bi-phase
- clock — external or internal
- baud rate — one of 4 per given MPU $\phi 2$ clock frequency or external clock X8 input
- wake-up feature — enabled or disabled
- interrupt requests — enabled or masked individually for transmitter and receiver data registers
- clock output — internal clock enabled or disabled to Port 2 (Bit 2)
- Port 2 (bits 3 and 4) — dedicated or not dedicated to serial I/O individually for transmitter and receiver.

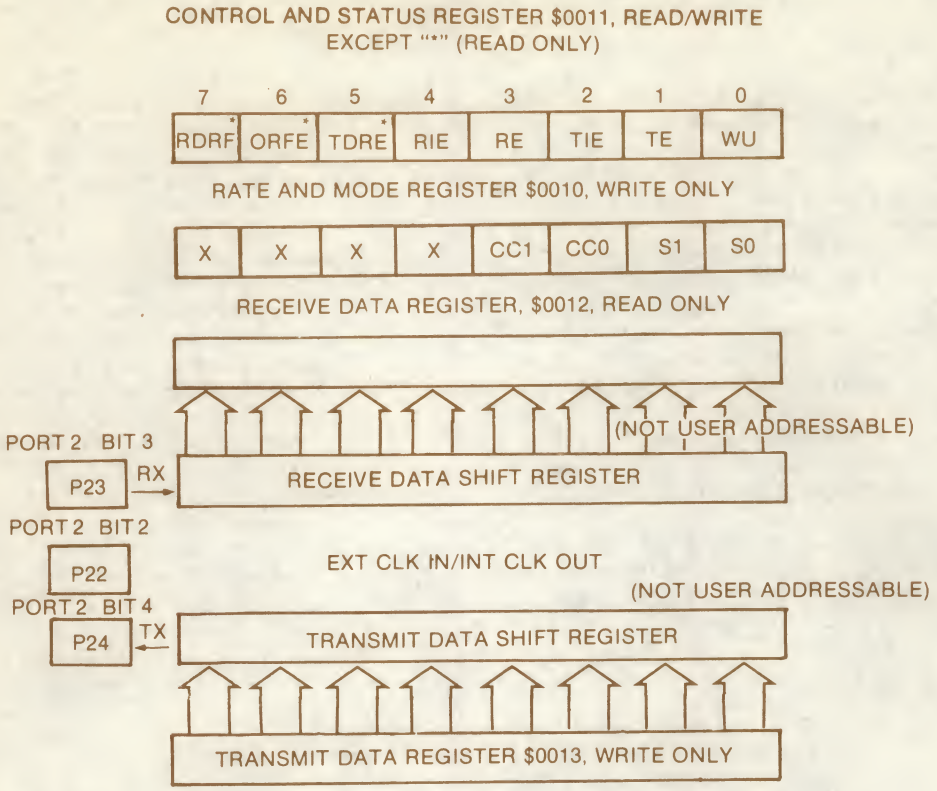
Serial Communications Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 23. The registers include:

- an 8-bit control and status register
- a 4-bit rate and mode control register (write only)
- an 8-bit read only receive data register and
- an 8-bit write only transmit data register.

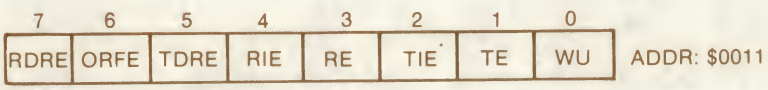
In addition to the four registers, the serial I/O section utilizes bit 3 (serial input) and bit 4 (serial output) or Port 2. Bit 2 of Port 2 is utilized if the internal-clock-out or external-clock-in options are selected.

FIGURE 23 — SERIAL I/O REGISTERS



Transmit/Receive Control and Status (TRCS) Register

The TRCS register consists of an 8-bit register of which all 8 bits may be read while only bits 0-4 may be written. The register is initialized to $20 on RESET. The bits in the TRCS register are defined as follows:



Bit 0 **WU** "Wake-up" on Next Message — set by MC6801 software cleared by hardware on receipt of ten consecutive 1's.

Bit 1 **TE** Transmit Enable — set by MC6801/MC68701 to produce preamble of nine consecutive 1's and to enable gating of transmitter output to Port 2, bit 4 regardless of the DDR value corresponding to this bit; when clear, serial I/O has no effect on Port 2 bit 4.

Bit 2 **TIE** Transmit Interrupt Enable — when set, will permit an $\overline{IRQ2}$ interrupt to occur when bit 5 (TDRE) is set; when clear, the TDRE value is masked from the bus.

Bit 3 **RE** Receiver Enable — when set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit; when clear, serial I/O has no effect on Port 2 bit 3.

Bit 4 **RIE** Receiver Interrpt Enable — when set, will permit an $\overline{IRQ2}$ interrupt to occur when bit 7 (RDRF) or bit 6 (OR) is set; when clear, the interrupt is masked.

Bit 5 **TDRE** Transmit Data Register Empty — set by hardware when a transfer is made from the transmit data register to the output shift register. The TDRE bit is cleared by reading the status register, then writing a new byte into the transmit data register, TDRE is initialized to 1 by $\overline{RESET}$.

Bit 6 **ORFE** Over-Run-Framing Error — set by hardware when an overrun or framing error occurs (receive only). An overrun is defined as a new byte received with last byte still in Data Register/Buffer. A framing error has occurred when the byte boundaries in bit stream are not synchronized to bit counter. The ORFE bit is cleared by reading the status register, then reading the Receive Data Register, or by $\overline{RESET}$.

Bit 7 **RDRF** Receiver Data Register Full — Set by hardware when a transfer from the input shift register to the receiver data register is made. The RDRF bit is cleared by reading the status register, then reading the Receive Data Register, or by $\overline{RESET}$.

Rate and Mode Control Register

The Rate and Mode Control register controls the following serial I/O variables:

- Baud rate
- format
- clocking source, and
- Port 2 bit 2 configuration

The register consists of 4 bits all of which are write-only and cleared on $\overline{RESET}$. The 4 bits in the register may be considered as a pair of 2-bit fields. The two low order bits control the bit rate for internal clocking and the remaining two bits control the format and clock select logic. The register definition is as follows:

7	6	5	4	3	2	1	0	
X	X	X	X	CC1	CC0	S1	S0	ADDR:$0010

Bit 0 **S0**
Bit 1 **S1**
Speed Select — These bits select the Baud rate for the internal clock. The four rates which may be selected are a function of the MPU $\phi 2$ clock frequency. Table 4 lists the available Baud rates.

Bit 2 **CC0**
Bit 3 **CC1**
Clock Control and Format Select — this 2-bit field controls the format and clock select logic. Table 5 defines the bit field.

TABLE 4 — SCI INTERNAL BAUD RATES

S1,S0	XTAL	4.0 MHz	4.9152 MHz	2.5476 MHz
	$\phi 2$	1.0 MHz	1.2288 MHz	0.6144 MHz
00	$\phi 2 \div 16$	62.5k Bits/s	76.8k Bits/s	38.4k Bits/s
01	$\phi 2 \div 128$	7,812.5 Bits/s	9,600 Bits/s	4,800 Bits/s
10	$\phi 2 \div 1024$	976.6 Bits/s	1,200 Bits/s	600 Bits/s
11	$\phi 2 \div 4096$	244.1 Bits/s	300 Bits/s	150 Bits/s

TABLE 5 — BIT FIELD

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	Bi-Phase	Internal	Not Used	**	**
01	NRZ	Internal	Not Used	**	**
10	NRZ	Internal	Output*	Serial Input	Serial Output
11	NRZ	External	Input	Serial Input	Serial Output

*Clock output is available regardless of values for bits RE and TE.
**Bit 3 is used for serial input if RE = "1" in TRCS; bit 4 is used for serial output if TE = "1" in TRCS.

Internally Generated Clock

If the user wishes for the serial I/O to furnish a clock, the following requirements are applicable:

the values of RE and TE are immaterial.

- the values of RE and TE are immaterial.
- CC1, CC0 must be set to 10
- the maximum clock rate will be $\phi \div 16$.
- the clock will be at 1X the bit rate and will have a rising edge at mid-bit.

Externally Generated Clock

If the user wishes to provide an external clock for the serial I/O, the following requirements are applicable:

- the CC1, CC0, field in the Rate and Mode Control Register must be set to 11,
- the external clock must be set to 8 times (X8) the desired baud rate and
- the maximum external clock frequency is 1.3 MHZ.

SERIAL OPERATIONS

The serial I/O hardware should be initialized by the MC6801 software prior to operation. This sequence will normally consist of:

- writing the desired operation control bits to the Rate and Mode Control Register and
- writing the desired operational control bits in the Transmit/Receive Control and Status Register.

The Transmitter Enable (TE) and Receiver Enable (RE) bits may be left set for dedicated operations.

Transmit Operations

The transmit operation is enabled by the TE bit in the Transmit/Receive Control and Status Register. This bit when set, gates the output of the serial transmit shift register to Port 2 Bit 4 and takes unconditional control over the Data Direction Register value for Port 2, Bit 4.

Following a $\overline{\text{RESET}}$, the user should configure both the Rate and Mode Control Register and the Transmit/Receiver Control and Status Register for desired operation. Setting the TE bit during tnis procedure initiates the serial output by first transmitting a ten-bit preamble of 1's. Following the preamble, internal synchronization is established and the transmitter section is ready for operation.

At this point one of two situations exist:

a) if the Transmit Data Register is empty (TDRE = 1), a continuous string of ones will be sent indicating an idle line, or

b) if data has been loaded into the Transmit Data Register (TDRE = 0), the word is transferred to the output shift register and transmission of the data word will begin.

During the transfer itself, the 0 start bit is first transmitted. Then the 8 data bits (beginning with bit 0) followed by the stop bit, are transmitted. When the Transmitter Data Register has been emptied, the hardware sets the TDRE flag bit.

If the MC6801 fails to respond to the flag within the proper time, (TDRE is still set when the next normal transfer from the parallel data register to the serial output register should occur) then a 1 will be sent (instead of a 0) at "Start" bit time, followed by more 1's until more data is supplied to the data register. No 0's will be sent while TDRE remains a 1.

The Bi-phase mode operates as described above except that the serial output toggles each bit time, and on 1/2 bit times when a 1 is sent.

Receive Operation

The receive operation is enabled by the RE bit which gates in the serial input through Port 2 Bit 3. The receiver section operation is conditioned by the contents of the Transmit/Receive Control and Status Register and the Rate and Mode Control Register.

The receiver bit interval is divided into 8 sub-intervals for internal synchronization. In the standard, non-Bi-phase mode, the received bit stream is synchronized by the first 0 (space) encountered.

The approximate center of each bit time is strobed during the next 10 bits. If the tenth bit is not a 1 (stop bit) a framing error is assumed, and bit ORFE is set. If the tenth bit is a 1, the data is transferred to the Receiver Data Register, and interrupt flag RDRF is set. If RDRF is still set at the next tenth bit time, ORFE will be set, indicating an over-run has occurred. When the MC6801 responds to either flag (RDRF or ORFE) by reading the status register followed by reading the Data Register, RDRF (or ORFE) will be cleared.

RAM CONTROL REGISTER

This register, which is addressed at $0014, gives status information about the standby RAM. A 0 in the RAM enable bit (RAM E) will disable the standby RAM, thereby protecting it at power down if V_{CC} is held greater than V_{SBB} volts, as explained previously in the signal description for V_{CC} Standby.

$0014	STANDBY BIT	RAME	X	X	X	X	X	X

Bit 0 Not Used.
Bit 1 Not Used.
Bit 2 Not used.
Bit 3 Not used.
Bit 4 Not used.
Bit 5 Not used.
Bit 6 The RAM ENABLE control bit allows the user the ability to disable the standby RAM. This bit is set to a logic "one" by reset which enables the standby RAM and can be written to one or zero under program control. When the RAM is disabled, logic "zero", data is read from external memory.
Bit 7 The STANDBY BIT of the control register, $0014, is cleared when the standby voltage is removed. This bit is a read/write status flag that the user can read which indicates that the standby RAM voltage has been applied, and the data in the standby RAM is valid.

FIGURE 24 — MEMORY MAP

The MC6801 provides up to 65k bytes of memory for program and/or data storage. The memory map is shown in Figure 24.

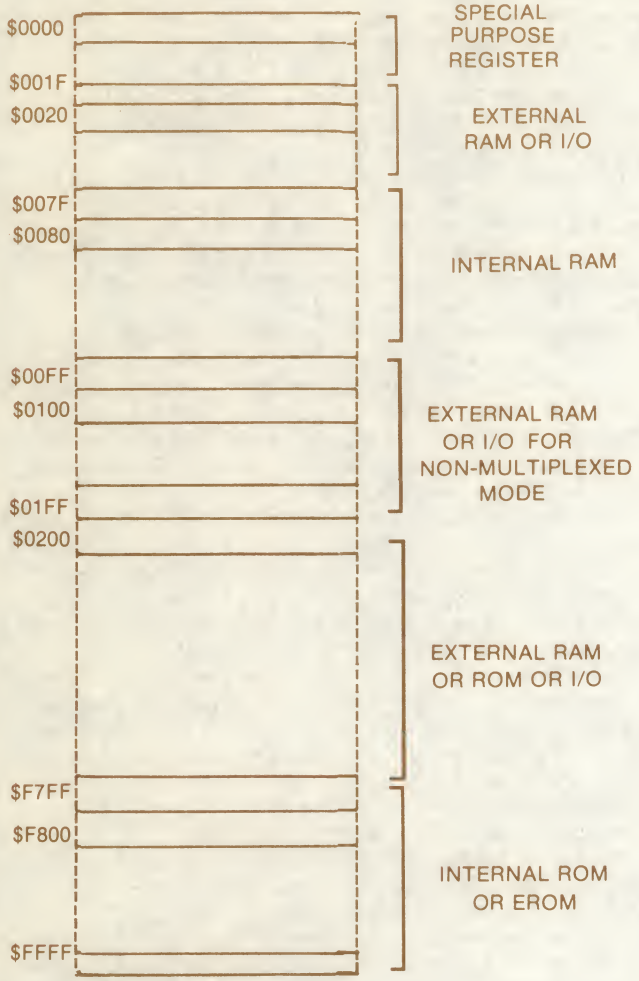


TABLE 6 — SPECIAL REGISTERS

The first 32 bytes are for the special purpose registers as shown in Table 6.

Hex Address	Register
00	Data Direction 1
01	Data Direction 2
02	I/O Port 1
03	I/O Port 2
04	Data Direction 3
05	Data Direction 4
06	I/O Port 3
07	I/O Port 4
08	TCSR
09	Counter High Byte
0A	Counter Low Byte
0B	Output Compare High Byte
0C	Output Compare Low Byte
0D	Input Capture High Byte
0E	Input Capture Low Byte
0F	I/O Port 3 C/S Register
10	Serial Rate and Mode Register
11	Serial Control and Status Register
12	Serial Receiver Data Register
13	Serial Transmit Data Register
14	RAM/EROM Control Register
15-1F	Reserved

FIGURE 25 — MEMORY MAP FOR INTERRUPT VECTORS

	Vector MS	Vector LS	Description
Highest Priority	FFFE,	FFFF	Restart
	FFFC,	FFFD	Non-Maskable Interrupt
	FFFA,	FFFB	Software Interrupt
	FFF8,	FFF9	IRQ1/Interrupt Strobe 3
	FFF6,	FFF7	IRQ2/Timer Input Capture
	FFF4,	FFF5	IRQ2/Timer Output Compare
	FFF2,	FFF3	IRQ2/Timer Overflow
Lowest Priority	FFF0,	FFF1	IRQ2/Serial I/O Interrupt

Locations $0020 through $007F access external RAM or I/O. Internal RAM is accessed at $0080 through $00FF. The RAM may be alternately selected by mask programming at location $A080. However, if the user desires to access external RAM at those locations he may do so by clearing the RAM ENABLE control bit of the RAM Control Register. In this way an extra 128 bytes of external RAM are available. The first 64 bytes of the 128 bytes of on-chip RAM are provided with a separate power supply. This will maintain the 64 bytes of RAM in the power down mode as explained in the pin description for Vcc Standby.

Locations $0100 through $01FF are available in the Expanded Non-Multiplexed Mode. The eight address lines of Port 4 make this 256 word expandability possible. Those not needed for address lines can be used as input lines instead.

The full range of addresses available to the user is in the Expanded Multiplexed Mode. Locations $0200 through $F7FF can be used as external RAM, external ROM, or I/O. Any higher order bits not required for addressing can be used as I/O as in the Expanded Non-Multiplexed Mode.

The internal ROM is located at $F800 through $FFFF. The decoder for the ROM may be mask programmed on A12, and A13 as zeros or one's to provide for $C800, $D800, $E800 for the ROM address. A12 and A13 may also be don't care in this decoder. The primary address for the ROM will be $F800.

GENERAL DESCRIPTION OF INSTRUCTION SET

The MC6801 is upward object code compatible with the MC6800 as it implements the full M6800 instruction set. The execution times of key instructions have been reduced to increase throughput. In addition, new instructions have been added; these include 16-bit operations and a hardware multiply.

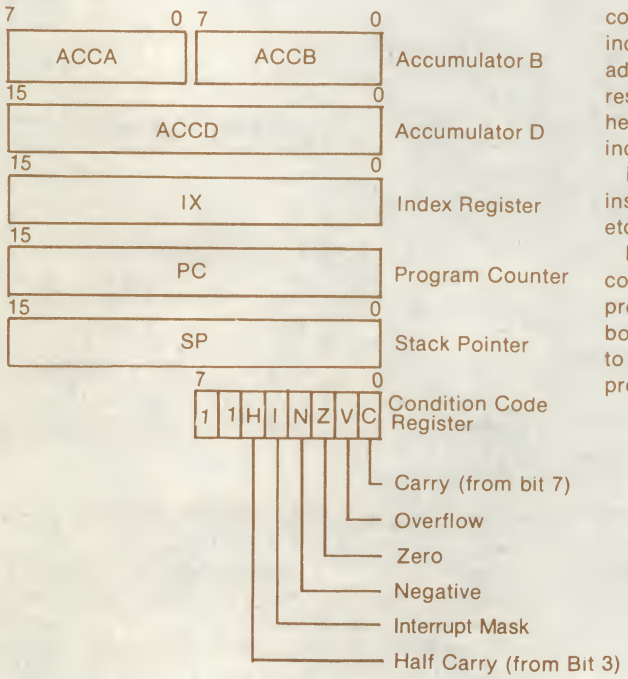
Included in the instruction set section are the following:

- MPU Programming Model (Figure 26)
- Addressing modes
- Accumulator and memory instructions — Table 7
- New instructions
- Index register and stack manipulations — Table 8
- Jump and branch instructions — Table 9
- Special operations — Figure 27
- Condition code register manipulation instructions — Table 10
- Instruction Execution times in machine cycles — Table 11
- Summary of cycle by cycle operation — Table 12

MPU PROGRAMMING MODEL

The programming model for the MC6801 is shown in Figure 26. The double (D) accumulator is physically the same as the A Accumulator concatenated with the B Accumulator so that any operation using accumulator D will destroy information in A and B.

FIGURE 26 — MCU PROGRAMMING MODEL



MPU ADDRESSING MODES

The MC6801 eight-bit microcomputer unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 11 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 4 MHz, these times would be microseconds.

Accumulator (ACCX) Addressing — In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing — In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The MCU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

Direct Addressing — In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing — In extended addressing, the address contained in the second byte of the instruction is used as the higher eight-bits of the address of the operand. The third byte of the instruction is used as the lower eight-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing — In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest eight bits in the MCU. The carry is then added to the higher order eight bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing — In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing — In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the high eight bits. This allows the user to address data within a range of -125 to +129 bytes of the present instruction. These are two-byte instructions.

TABLE 7—ACCUMULATOR & MEMORY INSTRUCTIONS

ACCUMULATOR AND MEMORY		ADDRESSING MODES																					
		IMMED.			DIRECT			INDEX			EXTEND			INHERENT			5	4	3	2	1	0
Operations	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	Boolean/Arithmetic Operation	H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				A:B+M:M+1 → A:B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A		•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B		•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B M	•	•	↕	↕	R	•
Clear	CLR							6F	6	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A - M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B - M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A - B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				M → M	•	•	↕	↕	R	S
	COMA													43	2	1	A → A	•	•	↕	↕	R	S
	COMB													53	2	1	B → B	•	•	↕	↕	R	S
Complement, 2's	NEG							60	6	2	70	6	3				OC - M → M	•	•	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 - A → A	•	•	↕	↕	①	②
NEGB														50	2	1	00 - B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M - 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A - 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B - 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDAD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M → A M + 1 → B	•	•	↕	↕	R	•

The Condition Code Register notes are listed after Table 10.

(Continued)

TABLE 7 — Continued

ACCUMULATOR AND MEMORY		ADDRESSING MODES																5	4	3	2	1	0
		IMMED.			DIRECT			INDEX			EXTEND			INHERENT									
Operations	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	Boolean/ Arithmetic Operation	H	I	N	Z	V	C
Multiply Unsigned	MUL													3D	10	1	A X B → A:B	•	•	•	•	•	(13)
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	3	1	A → M_{sp}, SP - 1 → SP	•	•	•	•	•	•
	PSHB													37	3	1	B → M_{sp}, SP - 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{sp} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{sp} → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3				M, A, B: C b_7 b_0	•	•	↕	↕	(6)	↕
	ROLA													49	2	1		•	•	↕	↕	(6)	↕
	ROLB													59	2	1		•	•	↕	↕	(6)	↕
Rotate Right	ROR							66	6	2	76	6	3				M, A, B: C b_7 b_0	•	•	↕	↕	(6)	↕
	RORA													46	2	1		•	•	↕	↕	(6)	↕
	RORB													56	2	1		•	•	↕	↕	(6)	↕
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M, A, B: C b_7 b_0 0	•	•	↕	↕	(6)	↕
	ASLA													48	2	1		•	•	↕	↕	(6)	↕
	ASLB													58	2	1		•	•	↕	↕	(6)	↕
Double Shift Left, Arithmetic	ASLD													05	3	1	C ← ACC A/ ACC B ← 0; A_7 A_0 B_7 B_0	•	•	↕	↕	(6)	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M, A, B: B_7 B_0 C	•	•	↕	↕	(6)	↕
	ASRA													47	2	1		•	•	↕	↕	(6)	↕
	ASRB													57	2	1		•	•	↕	↕	(6)	↕
Shift Right, Logical	LSR							64	6	2	74	6	3				M, A, B: 0 B7 B0 C	•	•	↕	↕	(6)	↕
	LSRA													44	2	1		•	•	↕	↕	(6)	↕
	LSRB													54	2	1				↕	↕	↕	↕
Double Shift Right Logical	LSRD													04	3	1	0 → ACC A/ ACCB → C; A7 A0 B7 B0	•	•	R	↕	(6)	↕
																		•	•	R	↕	6	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	•	•	↕	↕	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	•	•	↕	↕	R	•
Store Double Accumulator	STAD				DD	4	2	ED	5	2	FD	5	3				A → M; B → M + 1	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A - M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B - M → B	•	•	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				A:B - M:M + 1 → A:B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	A - B → A	•	•	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A - M - C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B - M - C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST							6D	6	2	7D	6	3				M - 00	•	•	↕	↕	RR	R
	TSTB													5D	2	1	B - 00	•	•	↕	↕	R	R

The Condition Code Register notes are listed after Table 10.

ADDED INSTRUCTIONS

In addition to the existing M6800 Instruction Set, the following new instructions are incorporated in the MC6801 Microcomputer.

ABX	Adds the 8-bit unsigned accumulator B to the 16-bit X-Register taking into account the possible carry out of the low order byte of the X-Register.	IX ← IX + ACCB
ADDD	Adds the double precision ACCD* to the double precision value M:M+1 and places the results in ACCD.	ACCD ← (ACCD) + (M:M+1)
ASLD	Shifts all bits of ACCAB one place to the left. Bit 0 is loaded with zero. The C bit is loaded from the most significant bit of ACCD.	
LDD	Loads the contents of double precision memory location into the double accumulator A:B. The condition codes are set according to the data.	ACCD ← (M:M+1)
LSRD	Shifts all bits of ACCD one place to the right. Bit 15 is loaded with zero. The C bit is loaded from the least significant bit to ACCD.	
MUL	Multiplies the 8 bits in accumulator A with the 8 bits in accumulator B to obtain a 16-bit unsigned number in A:B. ACCA contains MSB of result.	ACCD ← ACCA * ACCB
PSHX	The contents of the index register is pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by 2.	↓(IXL), SP ← (SP) - 1 ↓(IXL), SP ← (SP) - 1
PULX	The index register is pulled from the stack beginning at the current address contained in the stack pointer +1. The stack pointer is incremented by 2 in total.	SP ← (SP) + 1; IXH SP ← (SP) + 1; IHL
STD	Stores the contents of double accumulator A:B in memory. The contents of ACCD remain unchanged.	M:M + 1 ← (ACCD)
SUBD	Subtracts the contents of M:M + 1 from the contents of double accumulator AB and places the result in ACCD.	ACCAB ← (ACCD) - (M:M + 1)

*ACCD is the 16 bit register (A:B) formed by concatenating the A and B accumulators. The A-accumulator is the most significant byte.

TABLE 8 — INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG. 5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	4	3	9C	5	2	AC	6	2	BC	6	3				$X_H - M, X_L - (M + 1)$	•	•	⑦	↕	⑧	•
Decrement Index Reg	DEX													09	3	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	3	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	3	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	3	1	$1 SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Index Reg → Stack Pntr	TXS													35	3	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	3	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•
Add	ABX													3A	3	1	$B + X \rightarrow X$	•	•	•	•	•	•
Push Data	PSHX													3C	4	1	$X_L \rightarrow M_{SP}, SP - 1 \rightarrow SP$ $X_H \rightarrow H_{SP}, SP - 1 \rightarrow SP$	•	•	•	•	•	•
Pull Data	PULX													38	5	1	$SP + 1 \rightarrow SP, M_{SP} \rightarrow X_H$ $SP + 1 \rightarrow SP, M_{SP} \rightarrow X_L$	•	•	•	•	•	•

The Condition Code Register notes are listed after Table 10.

TABLE 9 — JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		———			(10)		———
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	(11)	•	•	•	•

TABLE 10 — CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	2	1	A → CCR	———			(12)		———
CCR → Accumulator A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

FIGURE 27 — SPECIAL OPERATIONS

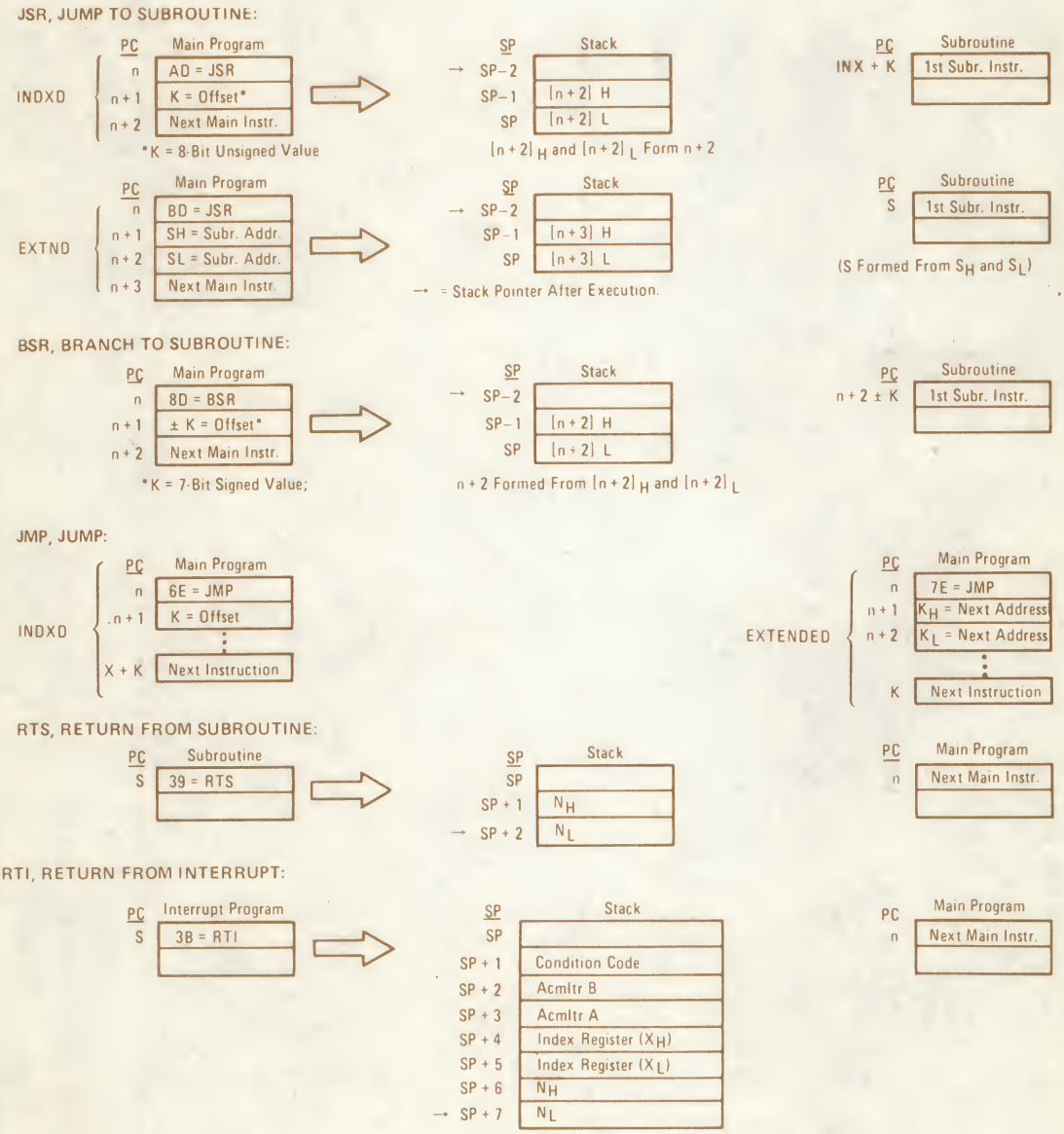


TABLE 11 — INSTRUCTION EXECUTION TIMES IN MACHINE CYCLE

	ACCX	Immediate	Direct	Extended	Indexed	Inherent	Relative
ABA	•	•	•	•	•	2	•
ABX	•	•	•	•	•	3	•
ADC	•	2	3	4	4	•	•
ADD	•	2	3	4	4	•	•
ADDD	•	4	5	6	6	•	•
AND	•	2	3	4	4	•	•
ASL	2	•	•	6	6	•	•
ASLD	•	•	•	•	•	3	•
ASR	2	•	•	6	6	•	•
BCC	•	•	•	•	•	•	3
BCS	•	•	•	•	•	•	3
BEQ	•	•	•	•	•	•	3
BGE	•	•	•	•	•	•	3
BGT	•	•	•	•	•	•	3
BHI	•	•	•	•	•	•	3
BIT	•	2	3	4	4	•	•
BLE	•	•	•	•	•	•	3
BLS	•	•	•	•	•	•	3
BLT	•	•	•	•	•	•	3
BMI	•	•	•	•	•	•	3
BNE	•	•	•	•	•	•	3
BPL	•	•	•	•	•	•	3
BRA	•	•	•	•	•	•	3
BSR	•	•	•	•	•	•	6
BVC	•	•	•	•	•	•	3
BVS	•	•	•	•	•	•	3
CBA	•	•	•	•	•	2	•
CLC	•	•	•	•	•	2	•
CLI	•	•	•	•	•	2	•
CLR	2	•	•	6	6	•	•
CLV	•	•	•	•	•	2	•
CMP	•	2	3	4	4	•	•
COM	2	•	•	6	6	•	•
CPX	•	4	5	6	6	•	•
DAA	•	•	•	•	•	2	•
DEC	2	•	•	6	6	•	•
DES	•	•	•	•	•	3	•
DEX	•	•	•	•	•	3	•
EOR	•	2	3	4	4	•	•
INC	2	•	•	6	6	•	•
INS	•	•	•	•	•	3	•

	ACCX	Immediate	Direct	Extended	Indexed	Inherent	Relative
INX	•	•	•	•	•	3	•
JMP	•	•	•	3	3	•	•
JSR	•	•	5	6	6	•	•
LDA	•	2	3	4	4	•	•
LDD	•	3	4	5	5	•	•
LDS	•	3	4	5	5	•	•
LDX	•	3	4	5	5	•	•
LSR	2	•	•	6	6	•	•
LSRD	•	•	•	•	•	3	•
MUL	•	•	•	•	•	10	•
NEG	2	•	•	6	6	•	•
NOP	•	•	•	•	•	2	•
ORA	•	2	3	4	4	•	•
PSH	3	•	•	•	•	•	•
PSHX	•	•	•	•	•	4	•
PUL	4	•	•	•	•	•	•
PULX	•	•	•	•	•	5	•
ROL	2	•	•	6	6	•	•
ROR	2	•	•	6	6	•	•
RTI	•	•	•	•	•	10	•
RTS	•	•	•	•	•	5	•
SBA	•	•	•	•	•	2	•
SBC	•	2	3	4	4	•	•
SEC	•	•	•	•	•	2	•
SEI	•	•	•	•	•	2	•
SEV	•	•	•	•	•	2	•
STA	•	•	3	4	4	•	•
STD	•	•	4	5	5	•	•
STS	•	•	4	5	5	•	•
STX	•	•	4	5	5	•	•
SUB	•	2	3	4	4	•	•
SUBD	•	4	5	6	6	•	•
SWI	•	•	•	•	•	12	•
TAB	•	•	•	•	•	2	•
TAP	•	•	•	•	•	2	•
TBA	•	•	•	•	•	2	•
TPA	•	•	•	•	•	2	•
TST	2	•	•	6	6	•	•
TSX	•	•	•	•	•	3	•
TXS	•	•	•	•	•	3	•
WAI	•	•	•	•	•	9	•

Summary of Cycle by Cycle Operation

Table 12 provides a detailed description of the information present on the Address Bus, Data Bus, and the Read/Write line (R/W) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware as the control program is executed. The information is categorized in groups according to addressing mode and number of cycles per instruction. (In general, instructions with the same addressing mode and number of cycles execute in the same manner; exceptions are indicated in the table).

TABLE 12 — CYCLE BY CYCLE OPERATION

ADDRESS MODE & INSTRUCTIONS	CYCLE	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
IMMEDIATE					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1 2	Op Code Address Op Code Address + 1	1 1	Op Code Operand Data
LDS LDX	3	1 2 3	Op Code Address Op Code Address + 1 Op Code Address + 2	1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte)
CPX SUBD ADDD	4	1 2 3 4	Op Code Address Op Code Address + 1 Op Code Address + 2 Address Bus FFFF	1 1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte) Low Byte of Restart Vector
DIRECT					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1 2 3	Op Code Address Op Code Address + 1 Address of Operand	1 1 1	Op Code Address of Operand Operand Data
STA	3	1 2 3	Op Code Address Op Code Address + 1 Destination Address	1 1 0	Op Code Destination Address Data from Accumulator
LDS LDX LDD	4	1 2 3 4	Op Code Address Op Code Address + 1 Address of Operand Operand Address + 1	1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte)
STS STX STD	4	1 2 3 4	Op Code Address Op Code Address + 1 Address of Operand Address of Operand + 1	1 1 0 0	Op Code Address of Operand Register Data (High Order Byte) Register Data (Low Order Byte)
CPX SUBD ADDD	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Operand Address Operand Address + 1 Address Bus FFFF	1 1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte) Low Byte of Restart Vector
JSR	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Subroutine Address Stack Pointer Stack Pointer + 1	1 1 1 0 0	Op Code Irrelevant Data First Subroutine Op Code Return Address (Low Order Byte) Return Address (High Order Byte)

(continued)

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
INDEXED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	1	Operand Data
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data
LDS LDX LDD	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	1	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STS STX STD	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
ASL LSR ASR NEG CLR ROL COM ROR DEC TST (1) INC	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	1	Current Operand Data
		5	Address Bus FFFF	1	Low Byte of Restart Vector
		6	Index Register Plus Offset	0	New Operand Data
CPX SUBD ADDD	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	Operand Data (High Order Byte)
		5	Index Register + Offset + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF		Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	First Subroutine Op Code
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer - 1	0	Return Address (High Order Byte)

(continued)

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
EXTENDED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Address of Operand
AND ORA		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
BIT SBC		4	Address of Operand	1	Operand Data
CMP SUB					
STA A	4	1	Op Code Address	1	Op Code
STA B		2	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	Operand Destination Address	0	Data from Accumulator
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
LDD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	1	Operand Data (High Order Byte)
		5	Address of Operand + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
STD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	0	Operand Data (High Order Byte)
		5	Address of Operand + 1	0	Operand Data (Low Order Byte)
ASL LSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
CLR ROL		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
COM ROR		4	Address of Operand	1	Current Operand Data
DEC TST (1)		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Address of Operand	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Operand Address (High Order Byte)
ADDD		3	Op code Address + 2	1	Operand Address (Low Order Byte)
		4	Operand Address	1	Operand Data (High Order Byte)
		5	Operand Address + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer - 1	0	Address of Operand (High Order Byte)

(continued)

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLES #	ADDRESS BUS	R/W LINE	DATA BUS
INHERENT		•			
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1 2	Op Code Address Op Code Address +1	1 1	Op Code Op Code of Next Instruction
ABX	3	1 2 3	Op Code Address Op Code Address +1 Address Bus FFFF	1 1 1	Op Code Irrelevent Data Low Byte of Restart Vector
ASLD LSRD	3	1 2 3	Op Code Address Op Code Address +1 Address Bus FFFF	1 1 1	Op Code Irrevelant Data Low Byte of Restart Vector
DES INS	3	1 2 3	Op Code Address Op Code Address +1 Previous Register Contents	1 1 1	Op Code Op Code of Next Instruction Irrelevant Data
INX DEX	3	1 2 3	Op Code Address Op Code Address +1 Address Bus FFFF	1 1 1	Op Code Op Code of Next Instruction Low Byte of Restart Vector
PSHA PSHB	3	1 2 3	Op Code Address Op Code Address +1 Stack Pointer	1 1 0	Op Code Op Code of Next Instruction Accumulator Data
ISX	3	1 2 3	Op Code Address Op Code Address +1 Stack Pointer	1 1 1	Op Code Op Code of Next Instruction Irrelevant Data
TXS	3	1 2 3	Op Code Address Op Code Address +1 Address Bus FFFF	1 1 1	Op Code Op Code of Next Instruction Low Byte of Restart Vector
PULA PULB	4	1 2 3 4	Op Code Address Op Code Address +1 Stack Pointer Stack Pointer +1	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data
PSHX	4	1 2 3 4	Op Code Address Op Code Address +1 Stack Pointer Stack Pointer -1	1 1 0 0	Op Code Irrelevant Data Index Register (Low Order Byte) Index Register (High Order Byte)
PULX 5	5	1 2 3 4 5	Op Code Address Op Code Address +1 Stack Pointer Stack Pointer +1 Stack Pointer +2	1 1 1 1 1	Op Code Irrelevant Data Irrelevant Data Index Register (High Order Byte) Index Register (Low Order Byte)
BCC BHT BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMT BVS	3	1 2 3	Op Code Address Op Code Address +1 Address Bus FFFF	1 1 1	Op Code Branch Offset Low Byte of Restart Vector
BSR	6	1 2 3 4 5 6	Op Code Address Op Code Address +1 Address Bus FFFF Subroutine Starting Address Stack Pointer Stack Pointer -1	1 1 1 1 0 0	Op Code Branch Offset Low Byte of Restart Vector Op Code of Next Instruction Return Address (Low Order Byte) Return Address (High Order Byte)

FIGURE 28 — MC6801 MCU SINGLE-CHIP DUAL PROCESSOR CONFIGURATION

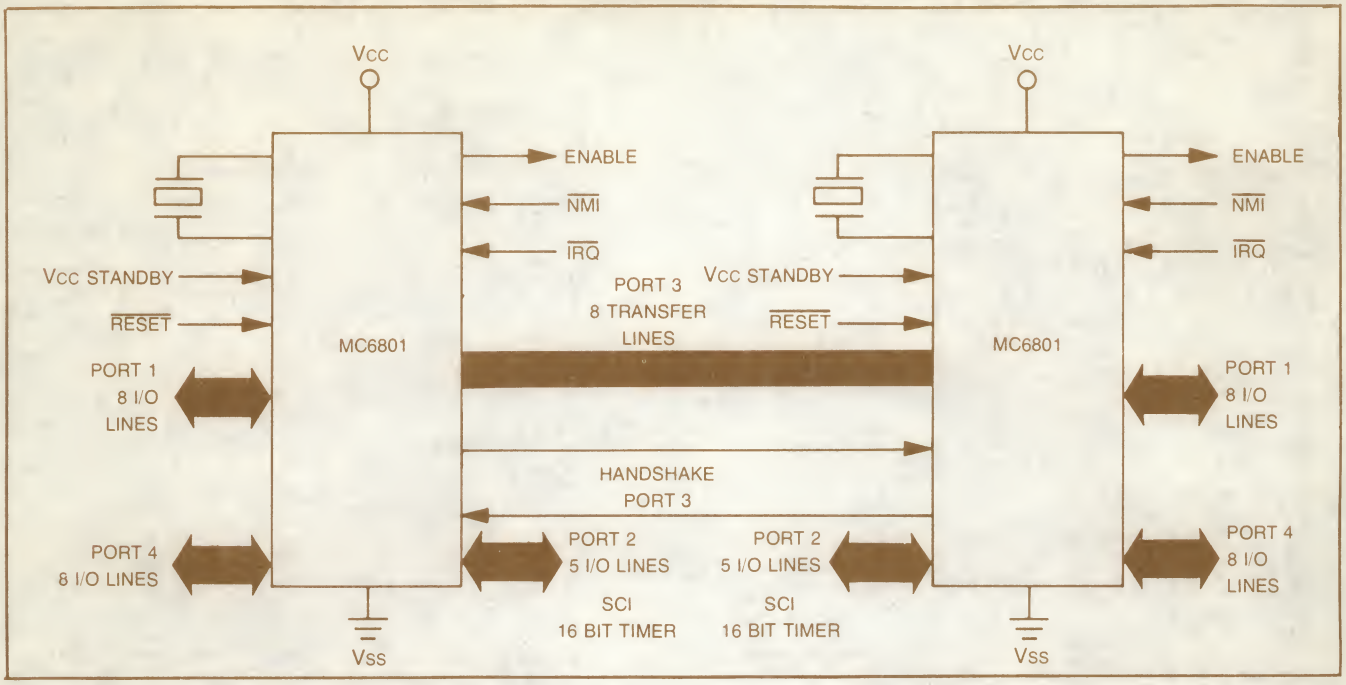


FIGURE 29 — MC6801 MCU
EXPANDED NON-MULTIPLEXED MODE

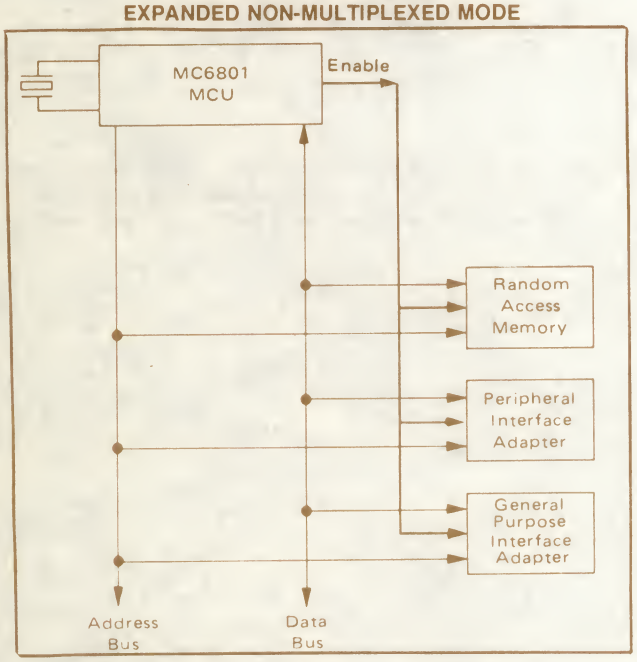
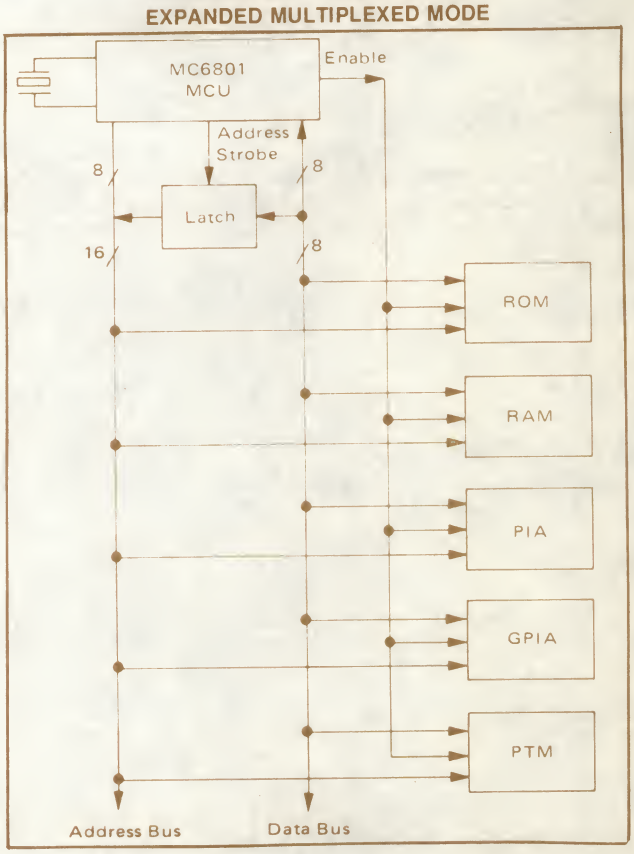
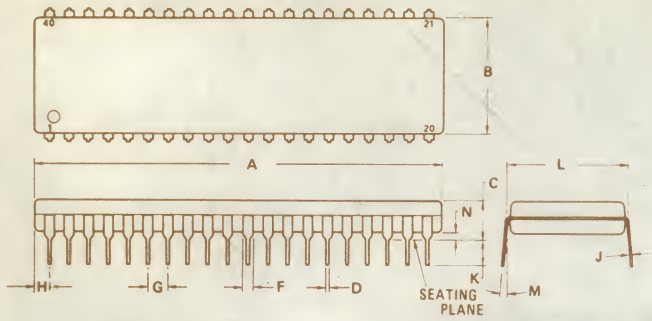


FIGURE 30 — MC6801 MCU
EXPANDED MULTIPLEXED MODE



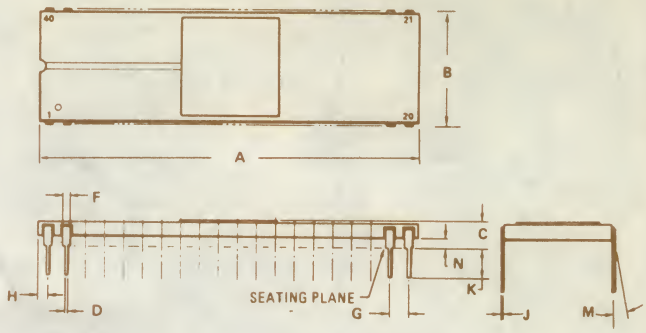
OUTLINE DIMENSIONS



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	50.29	51.31	1.980	2.020
B	14.86	15.62	0.585	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
M	0°	10°	0°	10°
N	0.51	1.52	0.020	0.060

L SUFFIX
CERAMIC PACKAGE
CASE 715-02

NOTE:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX. MAT'L CONDITION.



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.82	52.32	2.040	2.060
B	13.72	14.22	0.540	0.560
C	4.57	5.08	0.180	0.200
D	0.36	0.51	0.014	0.020
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.30	0.008	0.012
K	3.05	3.56	0.120	0.140
L	15.24 BSC		0.600 BSC	
M	0°	10°	0°	10°
N	0.51	1.02	0.020	0.040

P SUFFIX
PLASTIC PACKAGE
CASE 711-02

NOTES:
1. LEADS TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION (DIM "D").
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

MOTOROLA Semiconductor Products Inc
309-K78/5.0 Printed in Switzerland

MOTOROLA Semiconductors

MC6800
(1.0 MHz)
MC68A00
(1.5 MHz)
MC68B00
(2.0 MHz)

8-BIT MICROPROCESSING UNIT (MPU)

The MC6800 is a monolithic 8-bit microprocessor forming the central control function for Motorola's M6800 family. Compatible with TTL, the MC6800, as with all M6800 system parts, requires only one +5.0-volt power supply, and no external TTL devices for bus interface.

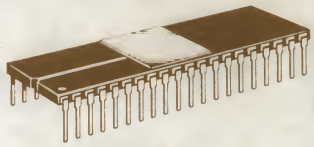
The MC6800 is capable of addressing 65K bytes of memory with its 16-bit address lines. The 8-bit data bus is bidirectional as well as 3-state, making direct memory addressing and multiprocessing applications realizable.

- Eight-Bit Parallel Processing
- Bidirectional Data Bus
- Sixteen-Bit Address Bus — 65K Bytes of Addressing
- 72 Instructions — Variable Length
- Seven Addressing Modes — Direct, Relative, Immediate, Indexed, Extended, Implied and Accumulator
- Variable Length Stack
- Vectored Restart
- Maskable Interrupt Vector
- Separate Non-Maskable Interrupt — Internal Registers Saved in Stack
- Six Internal Registers — Two Accumulators, Index Register, Program Counter, Stack Pointer and Condition Code Register
- Direct Memory Addressing (DMA) and Multiple Processor Capability
- Simplified Clocking Characteristics
- Clock Rates as High as 2.0 MHz
- Simple Bus Interface Without TTL
- Halt and Single Instruction Execution Capability

MOS

(N-CHANNEL, SILICON-GATE, DEPLETION LOAD)

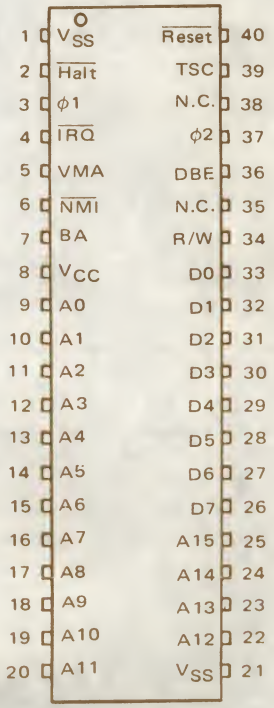
MICROPROCESSOR



L SUFFIX
CERAMIC PACKAGE
CASE 715

P SUFFIX
PLASTIC PACKAGE
CASE 711

PIN ASSIGNMENT



ORDERING INFORMATION

Speed	Device	Temperature Range
1.0 MHz	MC6800P, L	0 to 70°C
	MC6800CP, CL	−40 to +85°C
MIL-STD-883B MIL-STD-883C	MC6800BQCS MC6800CQCS	−55 to +125°C
1.5 MHz	MC68A00P, L	0 to +70°C
	MC68A00CP, CL	−40 to +85°C
2.0 MHz	MC68B00P, L	0 to +70°C

TABLE 1 — MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature Range—T_L to T_H MC6800, MC68A00, MC68B00 MC6800C, MC68A00C MC6800BQCS, MC6800CQCS	T_A	 0 to +70 -40 to +85 -55 to +125	°C
Storage Temperature Range	T_{stg}	-55 to +150	°C
Thermal Resistance Plastic Package Ceramic Package	θ_{JA}	 70 50	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

TABLE 2 — ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V, ± 5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic	V_{IH}	V_{SS} + 2.0	–	V_{CC}	Vdc
	$\phi1,\phi2$	V_{IHC}	V_{CC} – 0.6	–	V_{CC} + 0.3	
Input Low Voltage	Logic	V_{IL}	V_{SS} – 0.3	–	V_{SS} + 0.8	Vdc
	$\phi1,\phi2$	V_{ILC}	V_{SS} – 0.3	–	V_{SS} + 0.4	
Input Leakage Current		I_{in}				μAdc
(V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic*		–	1.0	2.5	
(V_{in} = 0 to 5.25 V, V_{CC} = 0.0 V)	$\phi1,\phi2$		–	–	100	
Three-State (Off State) Input Current	D0-D7	I_{TSI}	–	2.0	10	μAdc
(V_{in} = 0.4 to 2.4 V, V_{CC} = max)	A0-A15,R/$\overline{W}$		–	–	100	
Output High Voltage		V_{OH}				Vdc
(I_{Load} = -205 μAdc, V_{CC} = min)	D0-D7		V_{SS} + 2.4	–	–	
(I_{Load} = -145 μAdc, V_{CC} = min)	A0-A15,R/$\overline{W}$,VMA		V_{SS} + 2.4	–	–	
(I_{Load} = -100 μAdc, V_{CC} = min)	BA		V_{SS} + 2.4	–	–	
Output Low Voltage (I_{Load} = 1.6 mAdc, V_{CC} = min)		V_{OL}	–	–	V_{SS} + 0.4	Vdc
Power Dissipation		P_D	–	0.5	1.0	W
Capacitance		C_{in}				pF
(V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	$\phi1$		–	25	35	
	$\phi2$		–	45	70	
	D0-D7		–	10	12.5	
	Logic Inputs		–	6.5	10	
	A0-A15,R/$\overline{W}$,VMA	C_{out}	–	–	12	pF

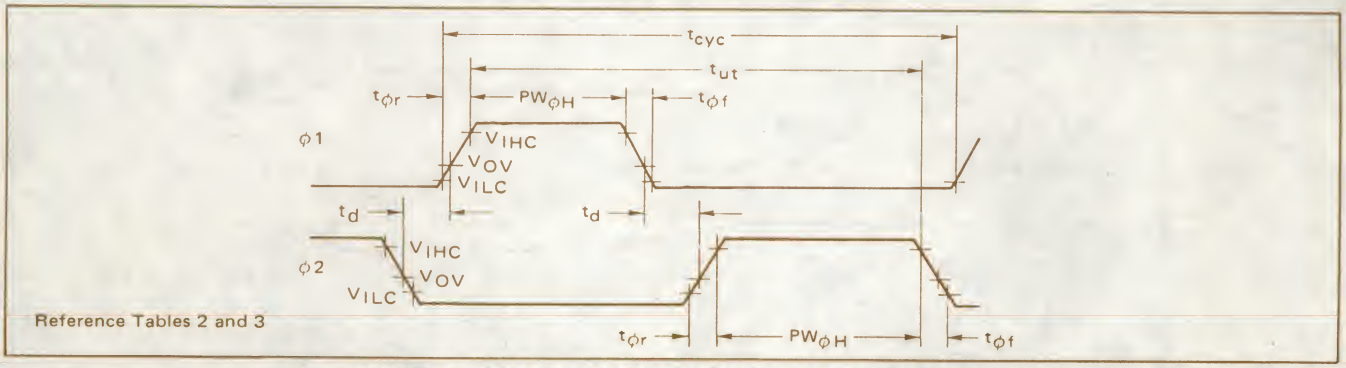
TABLE 3 — CLOCK TIMING (V_{CC} = 5.0 V, ±5%, V_{SS} = 0, T_A = T_L to T_H unless otherwise noted)

Characteristics		Symbol	Min	Typ	Max	Unit
Frequency of Operation	MC6800	f	0.1	–	1.0	MHz
	MC68A00		0.1	–	1.5	
	MC68B00		0.1	–	2.0	
Cycle Time (Figure 1)	MC6800	t_{cyc}	1.000	–	10	μs
	MC68A00		0.666	–	10	
	MC68B00		0.500	–	10	
Clock Pulse Width	$\phi1,\phi2$ – MC6800	$PW_{\phi H}$	400	–	9500	ns
(Measured at V_{CC} – 0.6 V)	$\phi1,\phi2$ – MC68A00		230	–	9500	
	$\phi1,\phi2$ – MC68B00		180	–	9500	
Total $\phi1$ and $\phi2$ Up Time	MC6800	t_{ut}	900	–	–	ns
	MC68A00		600	–	–	
	MC68B00		440	–	–	
Rise and Fall Times		$t_{\phi r}$, $t_{\phi f}$	–	–	100	ns
(Measured between V_{SS} + 0.4 and V_{CC} – 0.6)						
Delay Time or Clock Separation (Figure 1)		t_d				ns
(Measured at V_{OV} = V_{SS} + 0.6 V @ t_r = t_f ≤ 100 ns)			0	–	9100	
(Measured at V_{OV} = V_{SS} + 1.0 V @ t_r = t_f ≤ 35 ns)			0	–	9100	

TABLE 4 – READ/WRITE TIMING (Reference Figures 2 through 6)

Characteristic	Symbol	MC6800			MC68A00			MC68B00			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Address Delay	t_{AD}										ns
C = 90 pF		–	–	270	–	–	180	–	–	150	
C = 30 pF		–	–	250	–	–	165	–	–	135	
Peripheral Read Access Time $t_{ac} = t_{ut} - (t_{AD} + t_{DSR})$	t_{acc}	–	–	530	–	–	360	–	–	250	ns
Data Setup Time (Read)	t_{DSR}	100	–	–	60	–	–	40	–	–	ns
Input Data Hold Time	t_H	10	–	–	10	–	–	10	–	–	ns
Output Data Hold Time	t_H	10	25	–	10	25	–	10	25	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)	t_{AH}	30	50	–	30	50	–	30	50	–	ns
Enable High Time for DBE Input	t_{EH}	450	–	–	280	–	–	220	–	–	ns
Data Delay Time (Write)	t_{DDW}	–	–	225	–	–	200	–	–	160	ns
Processor Controls											
Processor Control Setup Time	t_{PCS}	200	–	–	140	–	–	110	–	–	ns
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}	–	–	100	–	–	100	–	–	100	ns
Bus Available Delay	t_{BA}	–	–	250	–	–	165	–	–	135	ns
Three-State Delay	t_{TSD}	–	–	270	–	–	270	–	–	220	ns
Data Bus Enable Down Time During $\phi 1$ Up Time	$t_{\overline{DBE}}$	150	–	–	120	–	–	75	–	–	ns
Data Bus Enable Rise and Fall Times	t_{DBEr}, t_{DBEf}	–	–	25	–	–	25	–	–	25	ns

FIGURE 1 – CLOCK TIMING WAVEFORM



Reference Tables 2 and 3

FIGURE 2 – READ DATA FROM MEMORY OR PERIPHERALS

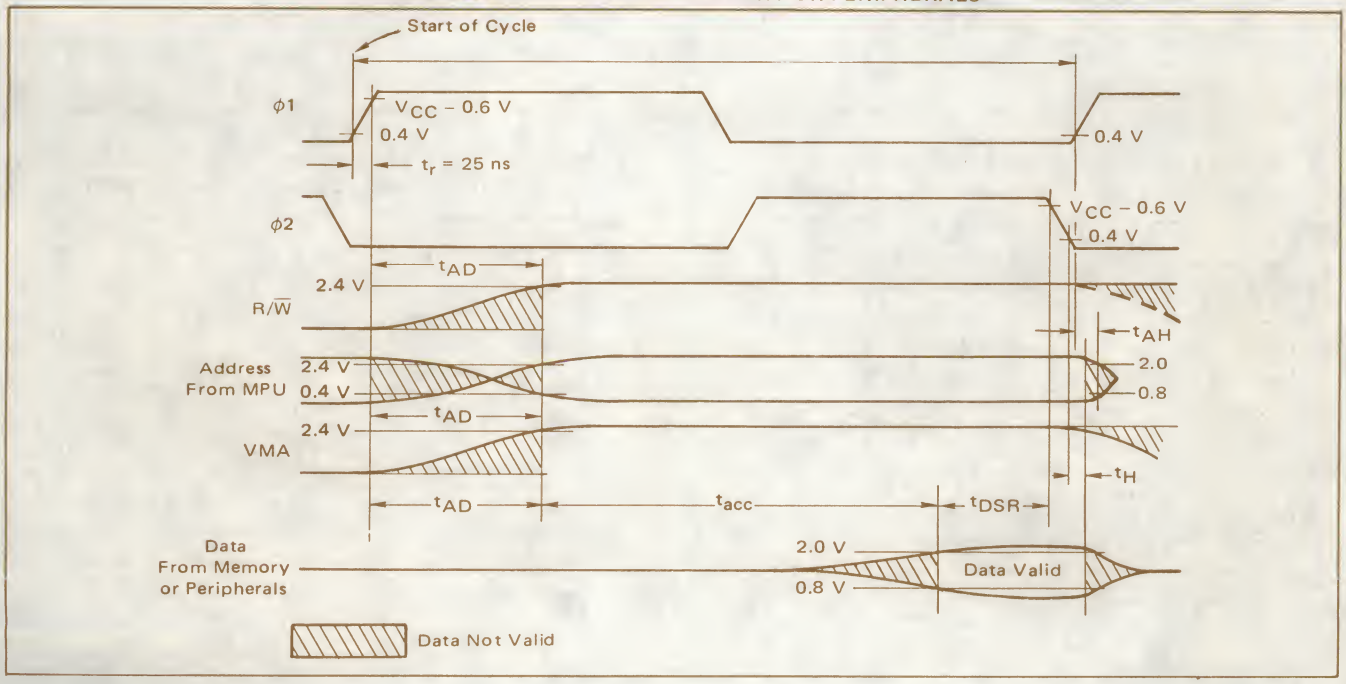


FIGURE 3 – WRITE IN MEMORY OR PERIPHERALS

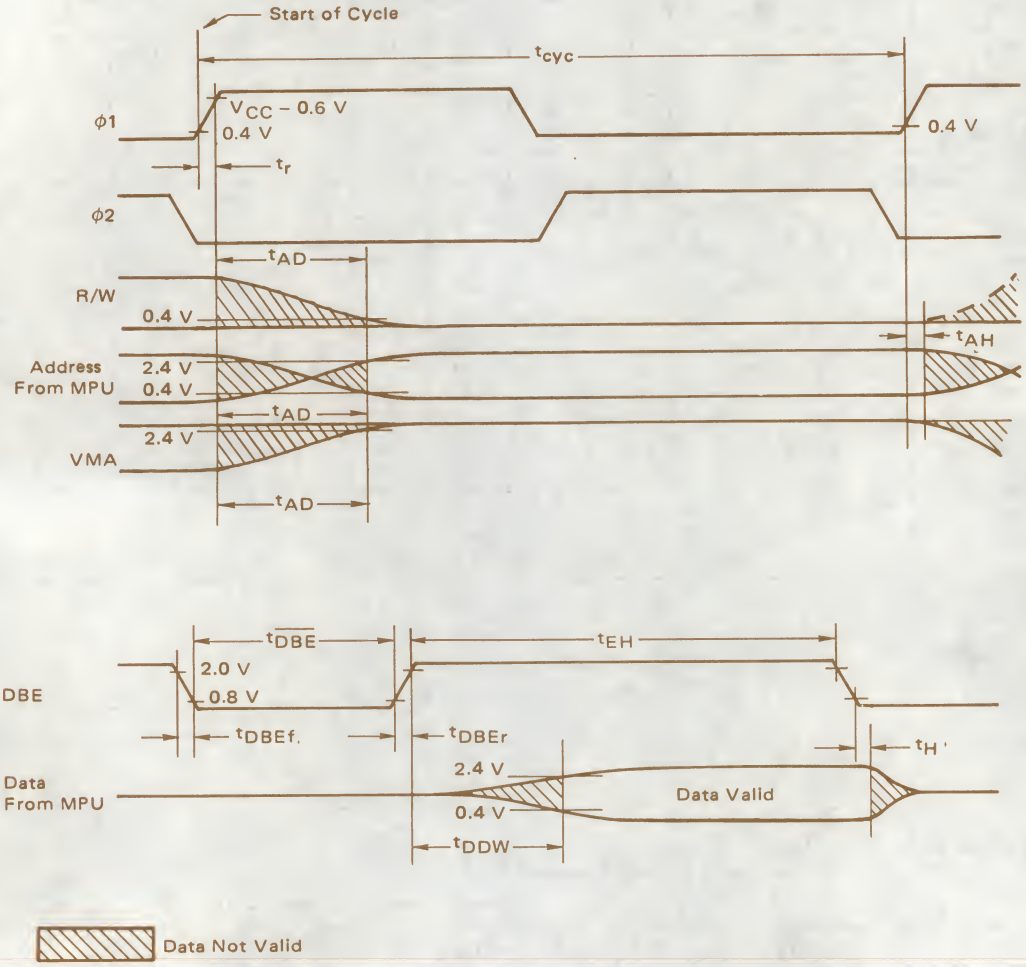


FIGURE 4 – TYPICAL DATA BUS OUTPUT DELAY versus CAPACITIVE LOADING (T_{DDW})

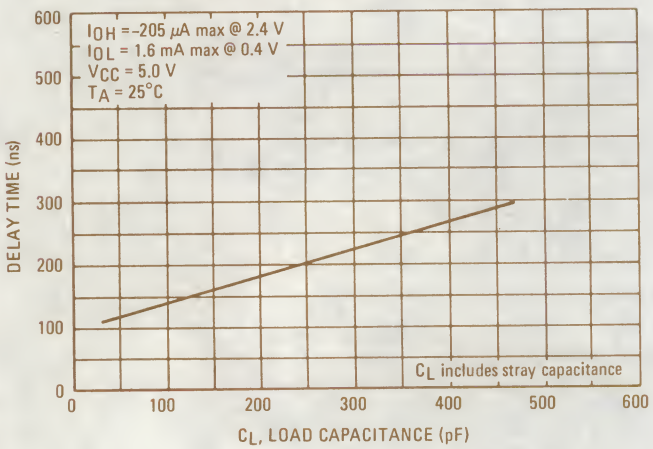


FIGURE 5 – TYPICAL READ/WRITE, VMA, AND ADDRESS OUTPUT DELAY versus CAPACITIVE LOADING (T_{AD})

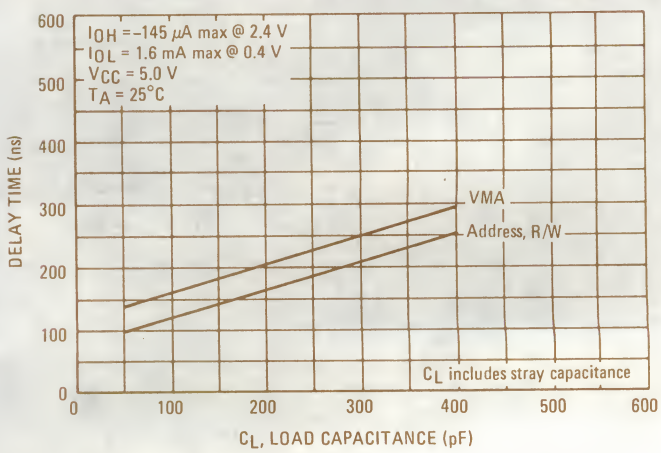
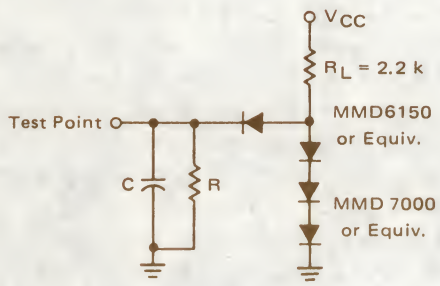


FIGURE 6 – BUS TIMING TEST LOADS



C = 130 pF for D0-D7, E
= 90 pF for A0-A15, R/W, and VMA (Except t_{AD2})
= 30 pF for A0-A15, R/W, and VMA (t_{AD2} only)
= 30 pF for BA

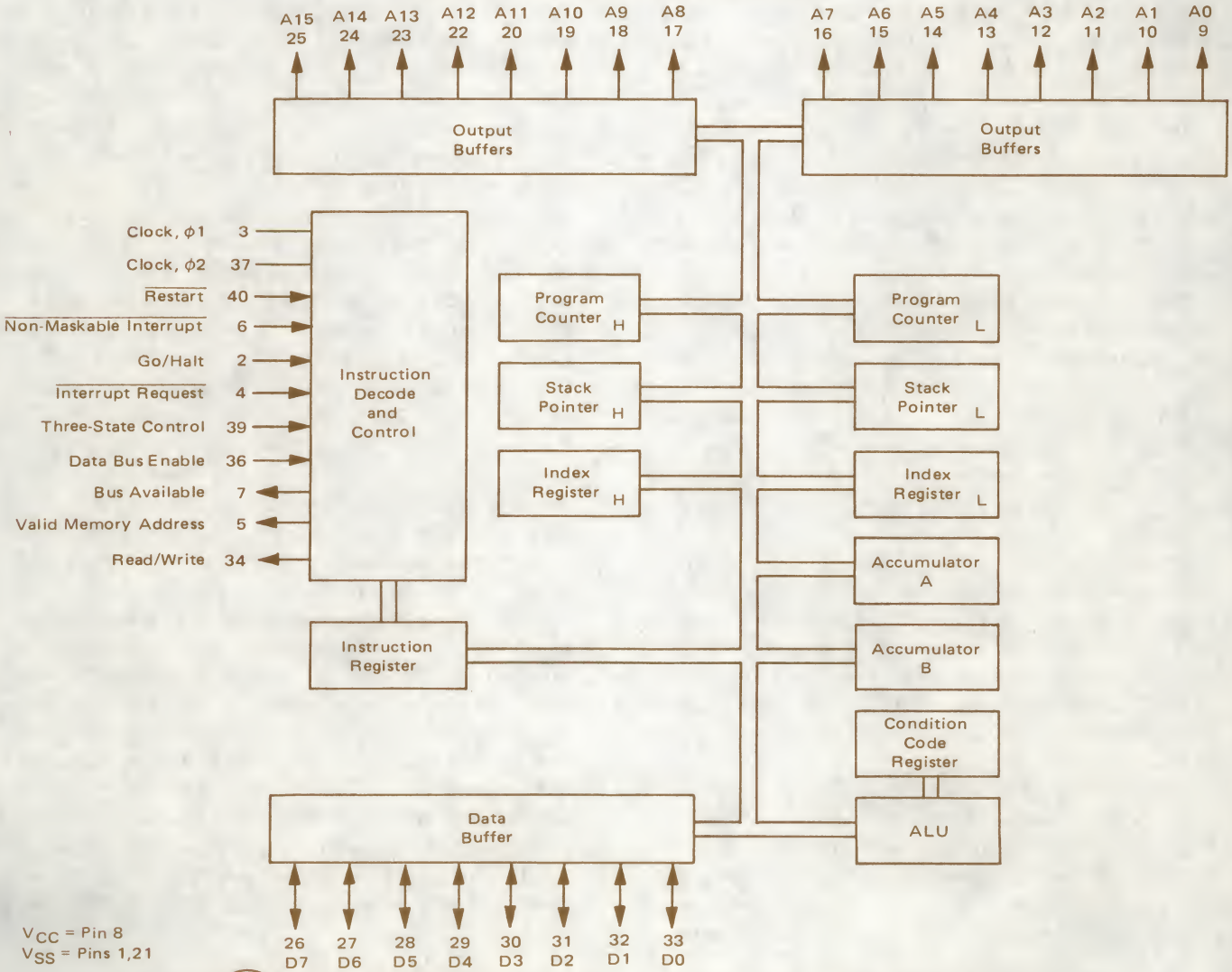
R = 11.7 kΩ for D0-D7
= 16.5 kΩ for A0-A15, R/W, and VMA
= 24 kΩ for BA

TEST CONDITIONS

The dynamic test load for the Data Bus is 130 pF and one standard TTL load as shown. The Address, R/W, and VMA outputs are tested under two conditions to allow optimum operation in both buffered and unbuffered systems. The resistor (R) is chosen to insure specified load currents during V_{OH} measurement.

Notice that the Data Bus lines, the Address lines, the Interrupt Request line, and the DBE line are all specified and tested to guarantee 0.4 V of dynamic noise immunity at both "1" and "0" logic levels.

FIGURE 7 – EXPANDED BLOCK DIAGRAM



V_{CC} = Pin 8
V_{SS} = Pins 1,21

MPU SIGNAL DESCRIPTION

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor.

Clocks Phase One and Phase Two (ϕ1,ϕ2) — Two pins are used for a two-phase non-overlapping clock that runs at the V_{CC} voltage level.

Figure 1 shows the microprocessor clocks, and Table 3 shows the static and dynamic clock specifications. The high level is specified at V_{IHC} and the low level is specified at V_{ILC}. The allowable clock frequency is specified by f (frequency). The minimum ϕ1 and ϕ2 high level pulse widths are specified by $PW_{\phi H}$ (pulse width high time). To guarantee the required access time for the peripherals, the clock up time, t_{ut}, is specified. Clock separation, t_d, is measured at a maximum voltage of V_{OV} (overlap voltage). This allows for a multitude of clock variations at the system frequency rate.

Address Bus (A0-A15) — Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load and 90 pF. When the output is turned off, it is essentially an open circuit. This permits the MPU to be used in DMA applications. Putting TSC in its high state forces the Address bus to go into the three-state mode.

Data Bus (D0-D7) — Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130 pF. Data Bus is placed in the three-state mode when DBE is low.

Data Bus Enable (DBE) — This input is the three-state control signal for the MPU data bus and will enable the bus drivers when in the high state. This input is TTL compatible; however in normal operation, it would be driven by the phase two clock. During an MPU read cycle, the data bus drivers will be disabled internally. When it is desired that another device control the data bus such as in Direct Memory Access (DMA) applications, DBE should be held low.

If additional data setup or hold time is required on an MPU write, the DBE down time can be decreased as shown in Figure 3 (DBE $\neq \phi$2). The minimum down time for DBE is $t_{\overline{DBE}}$ as shown and must occur within ϕ1 up time. The minimum delay from the trailing edge of DBE to the trailing edge of ϕ1 is t_{DBED}. By skewing DBE with respect to E in this manner, data setup or hold time can be increased.

Bus Available (BA) — The Bus Available signal will normally be in the low state; when activated, it will go to the high state indicating that the microprocessor has stopped and that the address bus is available. This will occur if the Halt line is in the low state or the processor is in the WAIT state as a result of the execution of a WAIT instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level. The processor is removed from the WAIT state by the occurrence of a maskable (mask bit I = 0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30 pF. If TSC is in the high state, Bus Available will be low.

Read/Write (R/$\overline{W}$) — This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read (high) or Write (low) state. The normal standby state of this signal is Read (high). Three-State Control going high will turn Read/Write to the off (high impedance) state. Also, when the processor is halted, it will be in the off state. This output is capable of driving one standard TTL load and 90 pF.

$\overline{\text{Reset}}$ — The $\overline{\text{Reset}}$ input is used to reset and start the MPU from a power down condition resulting from a power failure or initial start-up of the processor. This input can also be used to reinitialize the machine at any time after start-up.

If a high level is detected in this input, this will signal the MPU to begin the reset sequence. During the reset sequence, the contents of the last two locations (FFFE, FFFF) in memory will be loaded into the Program Counter to point to the beginning of the reset routine. During the reset routine, the interrupt mask bit is set and must be cleared under program control before the MPU can be interrupted by $\overline{\text{IRQ}}$. While $\overline{\text{Reset}}$ is low (assuming a minimum of 8 clock cycles have occurred) the MPU output signals will be in the following states: VMA = low, BA = low, Data Bus = high impedance, R/$\overline{W}$ = high (read state), and the Address Bus will contain the reset address FFFE. Figure 8 illustrates a power up sequence using the $\overline{\text{Reset}}$ control line. After the power supply reaches 4.75 V a minimum of eight clock cycles are required for the processor to stabilize in preparation for restarting. During these eight cycles, VMA will be in an indeterminate state so any devices that are enabled by VMA which could accept a false write during this time (such as a battery-backed RAM) must be disabled until VMA is forced low after eight cycles. $\overline{\text{Reset}}$ can go high asynchronously with the system clock any time after the eighth cycle.

$\overline{\text{Reset}}$ timing is shown in Figure 8 and Table 4. The maximum rise and fall transition times are specified by t_{PCr} and t_{PCf}. If $\overline{\text{Reset}}$ is high at t_{PCS} (processor control setup time) as shown in Figure 8 in any given cycle, then the restart sequence will begin on the next cycle as shown. The $\overline{\text{Reset}}$ control line may also be used to reinitialize the MPU system at any time during its operation. This is accomplished by pulsing $\overline{\text{Reset}}$ low for the duration of a minimum of three complete ϕ2 cycles. The $\overline{\text{Reset}}$ pulse can be completely asynchronous with the MPU system clock and will be recognized during ϕ2 if setup time t_{PCS} is met.

FIGURE 8 – $\overline{\text{RESET}}$ TIMING

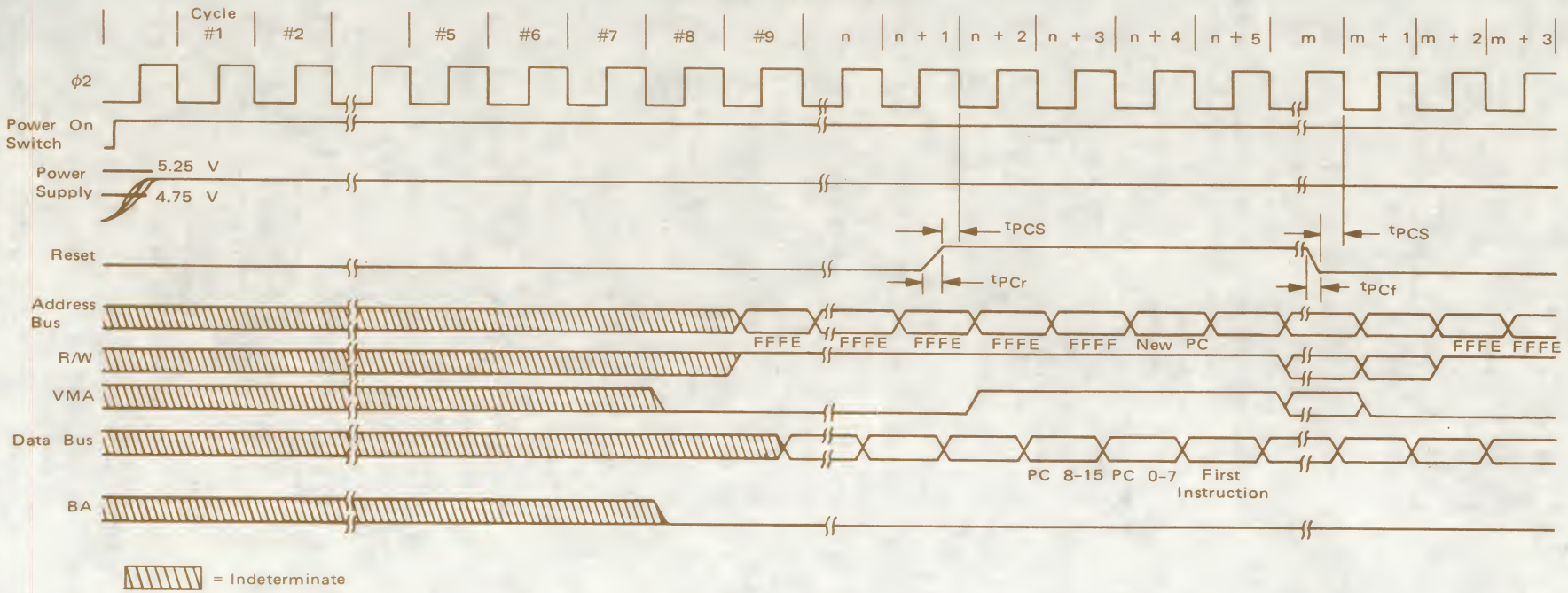
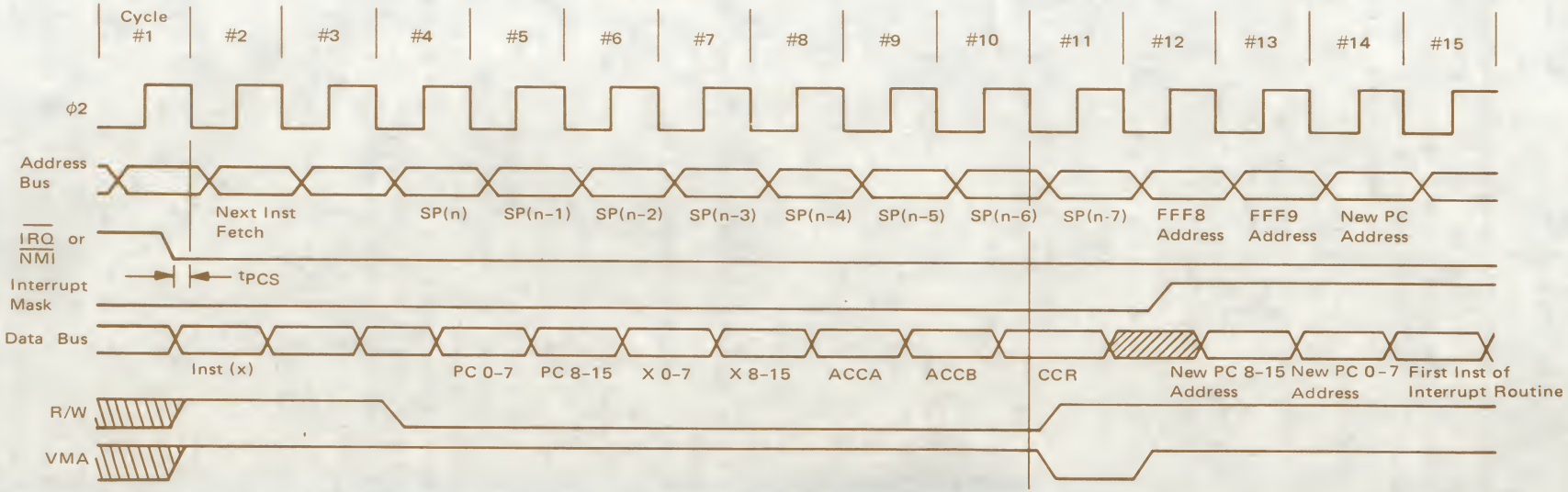


FIGURE 9 – INTERRUPT TIMING



Interrupt Request ($\overline{IRQ}$) – This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory. Interrupt timing is shown in Figure 9.

The $\overline{Halt}$ line must be in the high state for interrupts to be serviced. Interrupts will be latched internally while $\overline{Halt}$ is low.

The $\overline{IRQ}$ has a high impedance pullup device internal to the chip; however a 3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Non-Maskable Interrupt ($\overline{NMI}$) and Wait for Interrupt (WAI) – The MC6800 is capable of handling two types of interrupts: maskable ($\overline{IRQ}$) as described earlier, and non-maskable ($\overline{NMI}$). $\overline{IRQ}$ is maskable by the interrupt mask in the condition code register while $\overline{NMI}$ is not maskable. The handling of these interrupts by the MPU is the same except that each has its own vector address. The behavior of the MPU when interrupted is shown in Figure 9 which details the MPU response to an interrupt while the MPU is executing the control program. The interrupt shown could be either $\overline{IRQ}$ or $\overline{NMI}$ and can be asynchronous with respect to $\phi 2$. The interrupt is shown going low at time t_{PCS} in cycle #1 which precedes the first cycle of an instruction (OP code fetch). This instruction is not executed but instead the Program Counter (PC), Index Register (IX), Accumulators (ACCX), and the Condition Code Register (CCR) are pushed onto the stack.

The Interrupt Mask bit is set to prevent further interrupts. The address of the interrupt service routine is then fetched from FFFC, FFFD for an $\overline{NMI}$ interrupt and from FFF8, FFF9 for an $\overline{IRQ}$ interrupt. Upon completion of the interrupt service routine, the execution of RTI will pull the PC, IX, ACCX, and CCR off of the stack; the Interrupt Mask bit is restored to its condition prior to Interrupts.

Figure 11 is a similar interrupt sequence, except in this case, a WAIT instruction has been executed in preparation for the interrupt. This technique speeds up the MPU's response to the interrupt because the stacking of the PC, IX, ACCX, and the CCR is already done. While the MPU is waiting for the interrupt, Bus Available will go high indicating the following states of the control lines: VMA is low, and the Address Bus, R/$\overline{W}$ and Data Bus are all in the high impedance state. After the interrupt occurs, it is serviced as previously described.

TABLE 1 – MEMORY MAP FOR INTERRUPT VECTORS

Vector		
MS	LS	Description
FFFE	FFFF	Restart
FFFC	FFFD	Non-maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

Refer to Figure 11 for program flow for Interrupts.

Three State Control (TSC) – When the Three-State Control (TSC) line is a logic "1", the Address Bus and the R/$\overline{W}$ line are placed in a high impedance state. VMA and BA are forced low when TSC = "1" to prevent false reads or writes on any device enabled by VMA. It Is necessary to delay program execution while TSC is held high. This is done by insuring that no transitions of $\phi 1$ (or $\phi 2$) occur during this period. (Logic levels of the clocks are irrelevant so long as they do not change.) Since the MPU is a dynamic device, the $\phi 1$ clock can be stopped for a maximum time $PW_{\phi H}$ without destroying data within the MPU. TSC then can be used in a short Direct Memory Access (DMA) application.

Figure 12 shows the effect of TSC on the MPU. TSC must have its transitions at t_{TSE} (three-state enable) while holding $\phi 1$ high and $\phi 2$ low as shown. The Address Bus and R/$\overline{W}$ line will reach the high impedance state at t_{TSD} (three-state delay), with VMA being forced low. In this example, the Data Bus is also in the high impedance state while $\phi 2$ is being held low since DBE = $\phi 2$. At this point in time, a DMA transfer could occur on cycles #3 and #4. When TSC is returned low, the MPU Address and R/$\overline{W}$ lines return to the bus. Because it is too late in cycle #5 to access memory, this cycle is dead and used for synchronization. Program execution resumes in cycle #6.

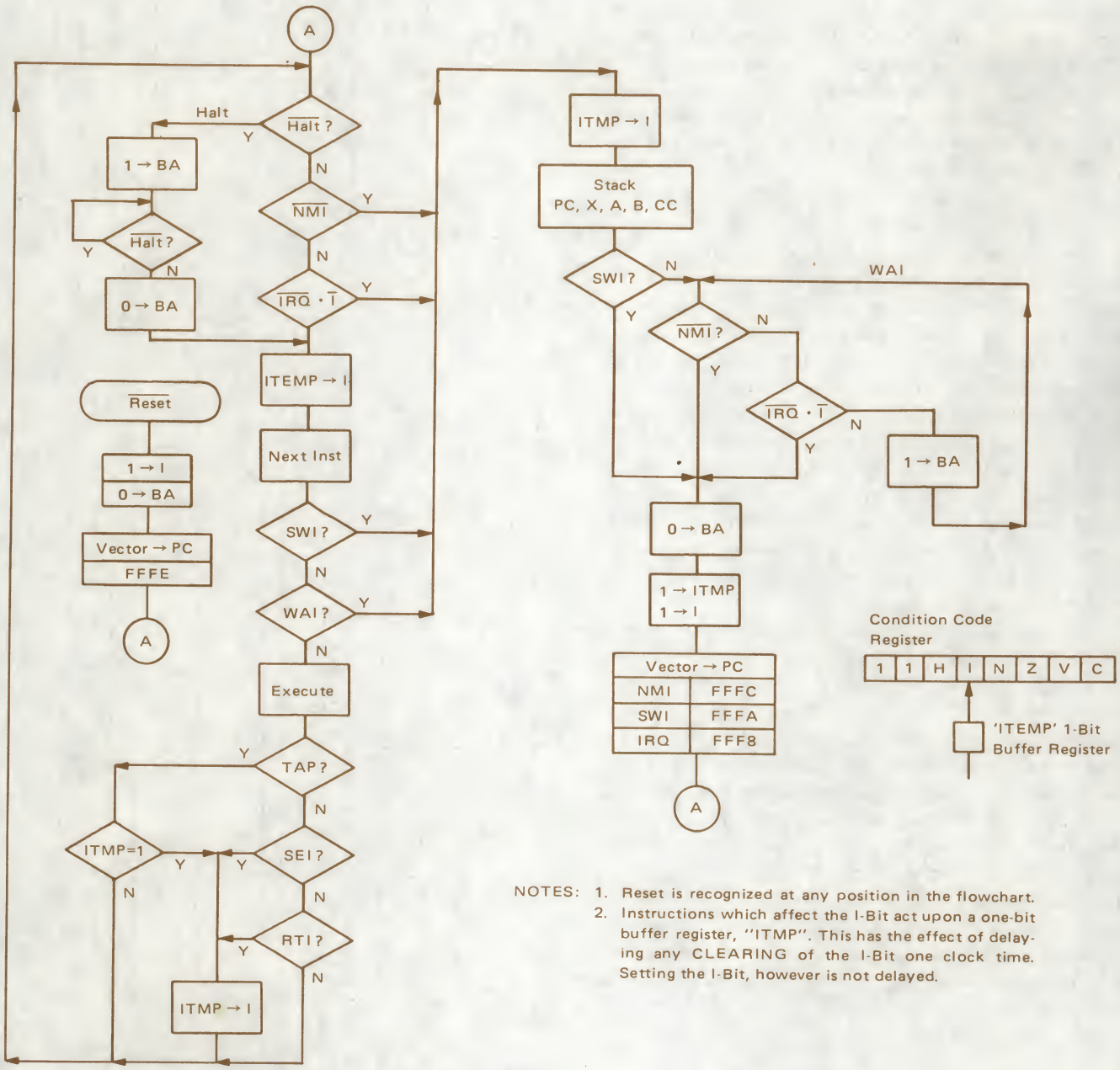
Valid Memory Address (VMA) – This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90 pF may be directly driven by this active high signal.

$\overline{Halt}$ – When this input is in the low state, all activity in the machine will be halted. This input is level sensitive.

The $\overline{Halt}$ line provides an input to the MPU to allow control of program execution by an outside source. If $\overline{Halt}$ is high, the MPU will execute the instructions; if it is low, the MPU will go to a halted or idle mode. A response signal, Bus Available (BA) provides an indication of the current MPU status. When BA is low, the MPU is in the process of executing the control program; if BA is high, the MPU has halted and all internal activity has stopped.

When BA is high, the Address Bus, Data Bus, and R/$\overline{W}$ line will be in a high impedance state, effectively removing the MPU from the system bus. VMA is forced low so that the floating system bus will not activate any device on the bus that is enabled by VMA.

FIGURE 10 – MPU FLOW CHART



NOTES: 1. Reset is recognized at any position in the flowchart.
2. Instructions which affect the I-Bit act upon a one-bit buffer register, "ITMP". This has the effect of delaying any CLEARING of the I-Bit one clock time. Setting the I-Bit, however is not delayed.

While the MPU is halted, all program activity is stopped, and if either an $\overline{NMI}$ or $\overline{IRQ}$ interrupt occurs, it will be latched into the MPU and acted on as soon as the MPU is taken out of the halted mode. If a $\overline{Reset}$ command occurs while the MPU is halted, the following states occur: VMA = low, BA = low, Data Bus = high impedance, R/$\overline{W}$ = high (read state), and the Address Bus will contain address FFFE as long as $\overline{Reset}$ is low. As soon as the Halt line goes high, the MPU will go to locations FFFE and FFFF for the address of the reset routine.

Figure13 shows the timing relationships involved when halting the MPU. The instruction illustrated is a one byte, 2 cycle instruction such as CLRA. When $\overline{Halt}$ goes low, the MPU will halt after completing execution of the current instruction. The transition of $\overline{Halt}$ must occur t_{PCS} before the trailing edge of $\phi 1$ of the last cycle of an instruction (point A of Figure 13). $\overline{Halt}$ must not go low any time later than the minimum t_{PCS} specified.

The fetch of the OP code by the MPU is the first cycle of the instruction. If $\overline{Halt}$ had not been low at Point A but went low during $\phi 2$ of that cycle, the MPU would have halted after completion of the following instruction. BA will go high by time t_{BA} (bus available delay time) after the last instruction cycle. At this point in time, VMA is low and R/$\overline{W}$, Address Bus, and the Data Bus are in the high impedance state.

To debug programs it is advantageous to step through programs instruction by instruction. To do this, $\overline{Halt}$ must be brought high for one MPU cycle and then returned low as shown at point B of Figure 13. Again, the transitions of $\overline{Halt}$ must occur t_{PCS} before the trailing edge of $\phi 1$. BA will go low at t_{BA} after the leading edge of the next $\phi 1$, indicating that the Address Bus, Data Bus, VMA and R/$\overline{W}$ lines are back on the bus. A single byte, 2 cycle instruction such as LSR is used for this example also. During the first cycle, the instruction Y is fetched from address M + 1. BA returns high at t_{BA} on the last cycle of the instruction indicating the MPU is off the bus. If instruction Y had been three cycles, the width of the BA low time would have been increased by one cycle.

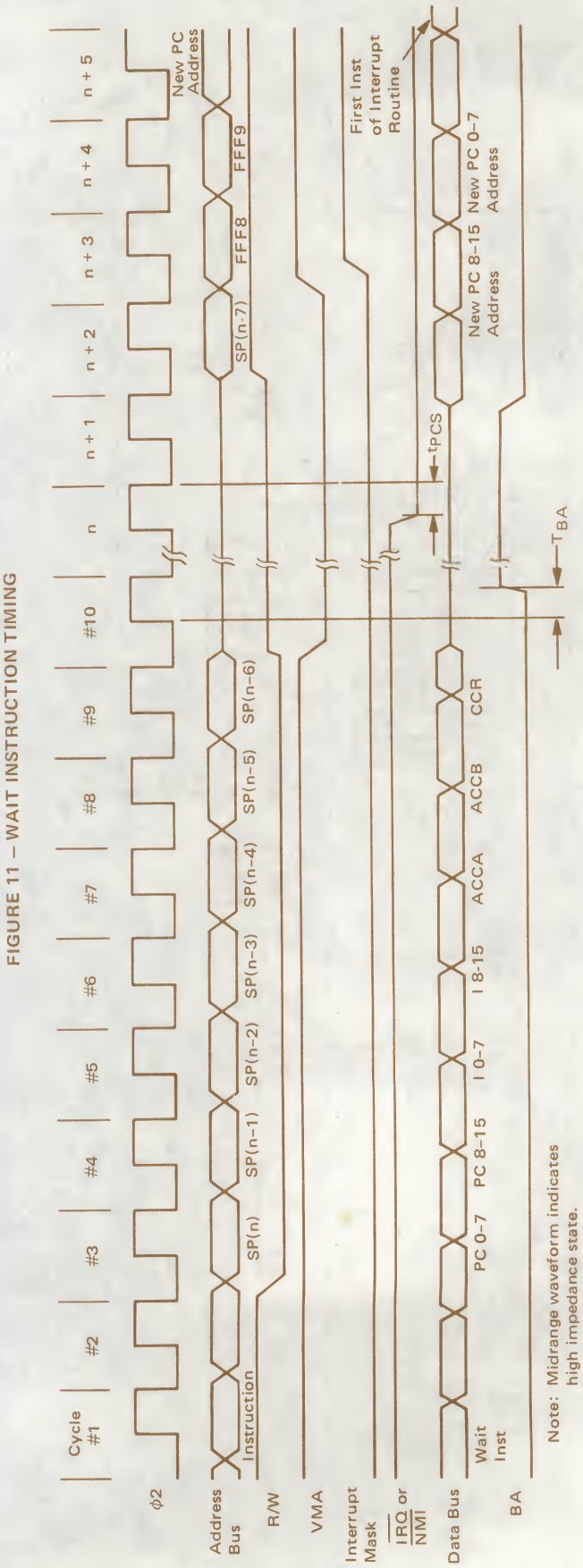


FIGURE 11 – WAIT INSTRUCTION TIMING

FIGURE 12 – THREE STATE CONTROL TIMING

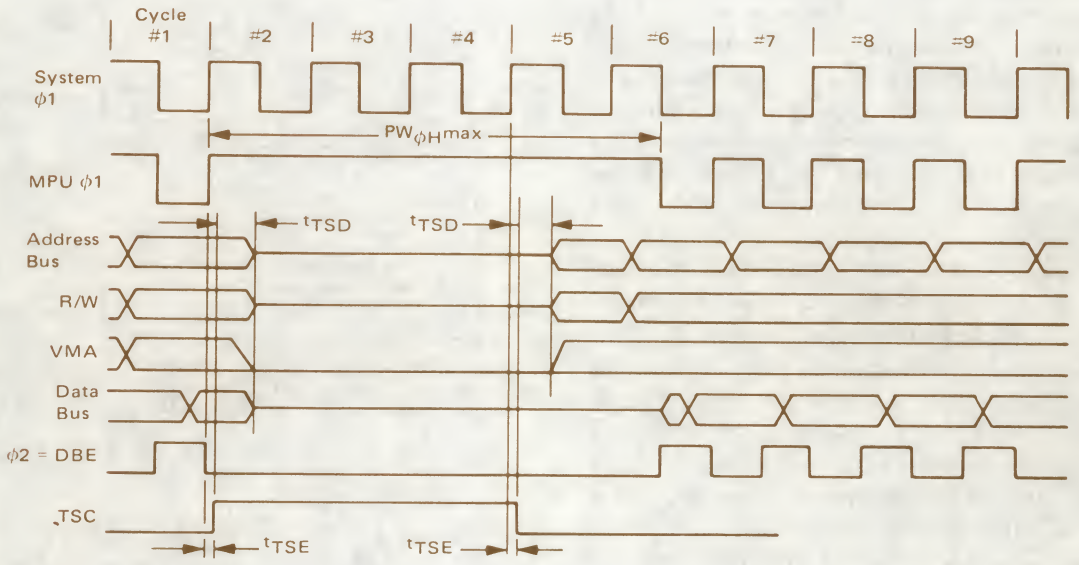
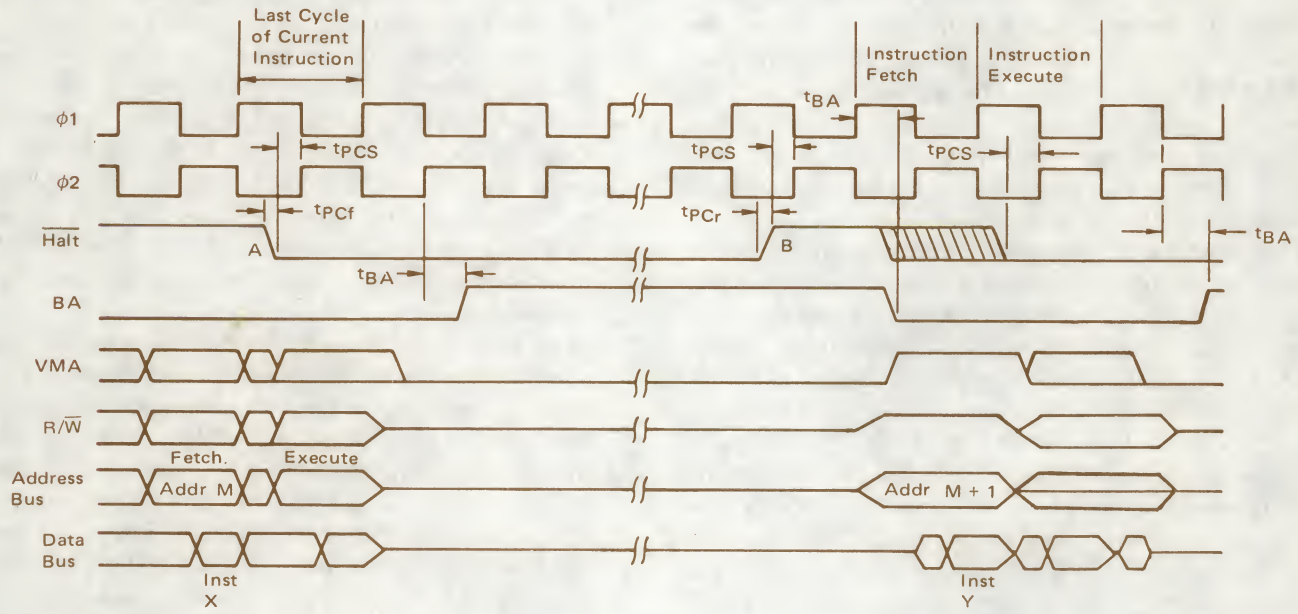


FIGURE 13 – $\overline{\text{HALT}}$ AND SINGLE INSTRUCTION EXECUTION FOR SYSTEM DEBUG



Note: Midrange waveform indicates high impedance state.

MPU REGISTERS

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Figure 14).

Program Counter – The program counter is a two byte (16 bits) register that points to the current program address.

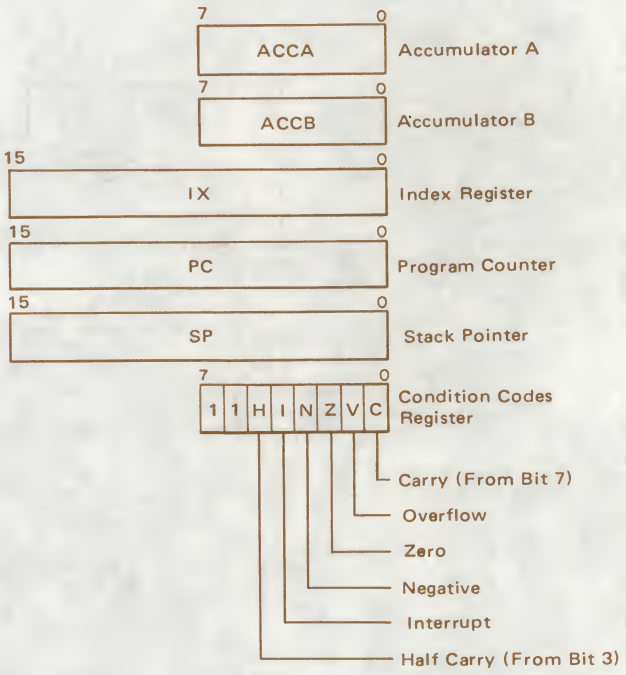
Stack Pointer – The stack pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be nonvolatile.

Index Register – The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

Accumulators – The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit (ALU).

Condition Code Register – The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and half carry from bit 3 (H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are ones.

FIGURE 14 – PROGRAMMING MODEL OF THE MICROPROCESSING UNIT



MPU INSTRUCTION SET

The MC6800 instructions are described in detail in the M6800 Programming Manual. This Section will provide a brief introduction and discuss their use in developing MC6800 control programs. The MC6800 has a set of 72 different executable source instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

Each of the 72 executable instructions of the source language assembles into 1 to 3 bytes of machine code. The number of bytes depends on the particular instruction and on the addressing mode. (The addressing modes which are available for use with the various executive instructions are discussed later.)

The coding of the first (or only) byte corresponding to an executable instruction is sufficient to identify the instruction and the addressing mode. The hexadecimal equivalents of the binary codes, which result from the translation of the 72 instructions in all valid modes of addressing, are shown in Table 6. There are 197 valid machine codes, 59 of the 256 possible codes being unassigned.

When an instruction translates into two or three bytes of code, the second byte, or the second and third bytes contain(s) an operand, an address, or information from which an address is obtained during execution.

Microprocessor instructions are often divided into three general classifications: (1) memory reference, so called because they operate on specific memory locations; (2) operating instructions that function without needing a memory reference; (3) I/O instructions for transferring data between the microprocessor and peripheral devices.

In many instances, the MC6800 performs the same operation on both its internal accumulators and the external memory locations. In additon, the MC6800 interfaces adapters (PIA and ACIA) allow the MPU to treat peripheral devices exactly like other memory locations, hence, no I/O instructions as such are required. Because of these features, other classifications are more suitable for introducing the MC6800's instruction set: (1) Accumulator and memory operations; (2) Program control operations; (3) Condition Code Register operations.

TABLE 6 – HEXADECIMAL VALUES OF MACHINE CODES

00	*			40	NEG	A		80	SUB	A	IMM	C0	SUB	B	IMM
01	NOP			41	*			81	CMP	A	IMM	C1	CMP	B	IMM
02	*			42	*			82	SBC	A	IMM	C2	SBC	B	IMM
03	*			43	COM	A		83	*			C3	*		
04	*			44	LSR	A		84	AND	A	IMM	C4	AND	B	IMM
05	*			45	*			85	BIT	A	IMM	C5	BIT	B	IMM
06	TAP			46	ROR	A		86	LDA	A	IMM	C6	LDA	B	IMM
07	TPA			47	ASR	A		87	*			C7	*		
08	INX			48	ASL	A		88	EOR	A	IMM	C8	EOR	B	IMM
09	DEX			49	ROL	A		89	ADC	A	IMM	C9	ADC	B	IMM
0A	CLV			4A	DEC	A		8A	ORA	A	IMM	CA	ORA	B	IMM
0B	SEV			4B	*			8B	ADD	A	IMM	CB	ADD	B	IMM
0C	CLC			4C	INC	A		8C	CPX	A	IMM	CC	*		
0D	SEC			4D	TST	A		8D	BSR		REL	CD	*		
0E	CLI			4E	*			8E	LDS		IMM	CE	LDX		IMM
0F	SEI			4F	CLR	A		8F	*			CF	*		
10	SBA			50	NEG	B		90	SUB	A	DIR	D0	SUB	B	DIR
11	CBA			51	*			91	CMP	A	DIR	D1	CMP	B	DIR
12	*			52	*			92	SBC	A	DIR	D2	SBC	B	DIR
13	*			53	COM	B		93	*			D3	*		
14	*			54	LSR	B		94	AND	A	DIR	D4	AND	B	DIR
15	*			55	*			95	BIT	A	DIR	D5	BIT	B	DIR
16	TAB			56	ROR	B		96	LDA	A	DIR	D6	LDA	B	DIR
17	TBA			57	ASR	B		97	STA	A	DIR	D7	STA	B	DIR
18	*			58	ASL	B		98	EOR	A	DIR	D8	EOR	B	DIR
19	DAA			59	ROL	B		99	ADC	A	DIR	D9	ADC	B	DIR
1A	*			5A	DEC	B		9A	ORA	A	DIR	DA	ORA	B	DIR
1B	ABA			5B	*			9B	ADD	A	DIR	DB	ADD	B	DIR
1C	*			5C	INC	B		9C	CPX		DIR	DC	*		
1D	*			5D	TST	B		9D	*			DD	*		
1E	*			5E	*			9E	LDS		DIR	DE	LDX		DIR
1F	*			5F	CLR	B		9F	STS		DIR	DF	STX		DIR
20	BRA		REL	60	NEG		IND	A0	SUB	A	IND	E0	SUB	B	IND
21	*			61	*			A1	CMP	A	IND	E1	CMP	B	IND
22	BHI		REL	62	*			A2	SBC	A	IND	E2	SBC	B	IND
23	BLS		REL	63	COM		IND	A3	*			E3	*		
24	BCC		REL	64	LSR		IND	A4	AND	A	IND	E4	AND	B	IND
25	BCS		REL	65	*			A5	BIT	A	IND	E5	BIT	B	IND
26	BNE		REL	66	ROR		IND	A6	LDA	A	IND	E6	LDA	B	IND
27	BEQ		REL	67	ASR		IND	A7	STA	A	IND	E7	STA	B	IND
28	BVC		REL	68	ASL		IND	A8	EOR	A	IND	E8	EOR	B	IND
29	BVS		REL	69	ROL		IND	A9	ADC	A	IND	E9	ADC	B	IND
2A	BPL		REL	6A	DEC		IND	AA	ORA	A	IND	EA	ORA	B	IND
2B	BMI		REL	6B	*			AB	ADD	A	IND	EB	ADD	B	IND
2C	BGE		REL	6C	INC		IND	AC	CPX		IND	EC	*		
2D	BLT		REL	6D	TST		IND	AD	JSR		IND	ED	*		
2E	BGT		REL	6E	JMP		IND	AE	LDS		IND	EE	LDX		IND
2F	BLE		REL	6F	CLR		IND	AF	STS		IND	EF	STX		IND
30	TSX			70	NEG		EXT	B0	SUB	A	EXT	F0	SUB	B	EXT
31	INS			71	*			B1	CMP	A	EXT	F1	CMP	B	EXT
32	PUL	A		72	*			B2	SBC	A	EXT	F2	SBC	B	EXT
33	PUL	B		73	COM		EXT	B3	*			F3	*		
34	DES			74	LSR		EXT	B4	AND	A	EXT	F4	AND	B	EXT
35	TXS			75	*			B5	BIT	A	EXT	F5	BIT	B	EXT
36	PSH	A		76	ROR		EXT	B6	LDA	A	EXT	F6	LDA	B	EXT
37	PSH	B		77	ASR		EXT	B7	STA	A	EXT	F7	STA	B	EXT
38	*			78	ASL		EXT	B8	EOR	A	EXT	F8	ADC	B	EXT
39	RTS			79	ROL		EXT	B9	ADC	A	EXT	F9	ADC	B	EXT
3A	*			7A	DEC		EXT	BA	ORA	A	EXT	FA	ORA	B	EXT
3B	RTI			7B	*			BB	ADD	A	EXT	FB	ADD	B	EXT
3C	*			7C	INC		EXT	BC	CPX		EXT	FC	*		
3D	*			7D	TST		EXT	BD	JSR		EXT	FD	*		
3E	WAI			7E	JMP		EXT	BE	LDS		EXT	FE	LDX		EXT
3F	SWI			7F	CLR		EXT	BF	STS		EXT	FF	STX		EXT

Notes: 1. Addressing Modes: A = Accumulator A, B = Accumulator B, REL = Relative, IND = Indexed, IMM = Immediate, DIR = Direct

2. Unassigned code indicated by "*"

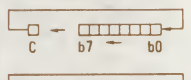
TABLE 7 – ACCUMULATOR AND MEMORY OPERATIONS

The accumulator and memory operations and their effect on the CCR are shown in Table 7. Included are Arithmetic Logic, Data Test and Data Handling instructions.

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	3B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

ADDRESSING MODES

COND. CODE REG.

⑥ 

LEGEND:

OP	Operation Code (Hexadecimal);	+	Boolean Inclusive OR;
~	Number of MPU Cycles;	⊕	Boolean Exclusive OR;
#	Number of Program Bytes;	M	Complement of M;
+	Arithmetic Plus;	→	Transfer Into;
−	Arithmetic Minus;	0	Bit = Zero;
·	Boolean AND;	00	Byte = Zero;
M_{SP}	Contents of memory location pointed to be Stack Pointer;		

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H	Half-carry from bit 3;
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

TABLE 7 – CONTINUED

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

1	(Bit V)	Test: Result = 10000000?
2	(Bit C)	Test: Result = 00000000?
3	(Bit C)	Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4	(Bit V)	Test: Operand = 10000000 prior to execution?
5	(Bit V)	Test: Operand = 01111111 prior to execution?
6	(Bit V)	Test: Set equal to result of N⊕C after shift has occurred.

PROGRAM CONTROL OPERATIONS

Program Control operation can be subdivided into two categories: (1) Index Register/Stack Pointer instructions; (2) Jump and Branch operations.

Index Register/Stack Pointer Operations

The instructions for direct operation on the MPU's Index Register and Stack Pointer are summarized in Table 8. Decrement (DEX, DES), increment (INX, INS), load (LDX, LDS), and store (STX, STS) instructions are provided for both. The Compare instruction, CPX, can be used to compare the Index Register to a 16-bit value and update the Condition Code Register accordingly.

The TSX instruction causes the Index Register to be loaded with the address of the last data byte put onto the "stack". The TXS instruction loads the Stack Pointer with a value equal to one less than the current contents of the Index Register. This causes the next byte to be pulled from the "stack" to come from the location indicated by the Index Register. The utility of these two instructions can be clarified by describing the "stack" concept relative to the M6800 system.

The "stack" can be thought of as a sequential list of data stored in the MPU's read/write memory. The Stack Pointer contains a 16-bit memory address that is used to access the list from one end on a last-in-first-out (LIFO) basis in contrast to the random access mode used by the MPU's other addressing modes.

The M6800 instruction set and interrupt structure allow extensive use of the stack concept for efficient handling of data movement, subroutines and interrupts. The instructions can be used to establish one or more "stacks" anywhere in read/write memory. Stack length is limited only by the amount of memory that is made available.

TABLE 8 – INDEX REGISTER AND STACK POINTER INSTRUCTIONS

		IMMED			DIRECT			INDEX			EXTND			IMPLIED				COND. CODE REG.					
POINTER OPERATIONS	MNEMONIC	OP	~	=	OP	~	=	OP	~	=	OP	~	=	OP	~	=	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	①	↕	②	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	③	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	③	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	③	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	③	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

① (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?
② (Bit V) Test: 2's complement overflow from subtraction of ms bytes?
③ (Bit N) Test: Result less than zero? (Bit 15 = 1)

FIGURE 15 – STACK OPERATION, PUSH INSTRUCTION

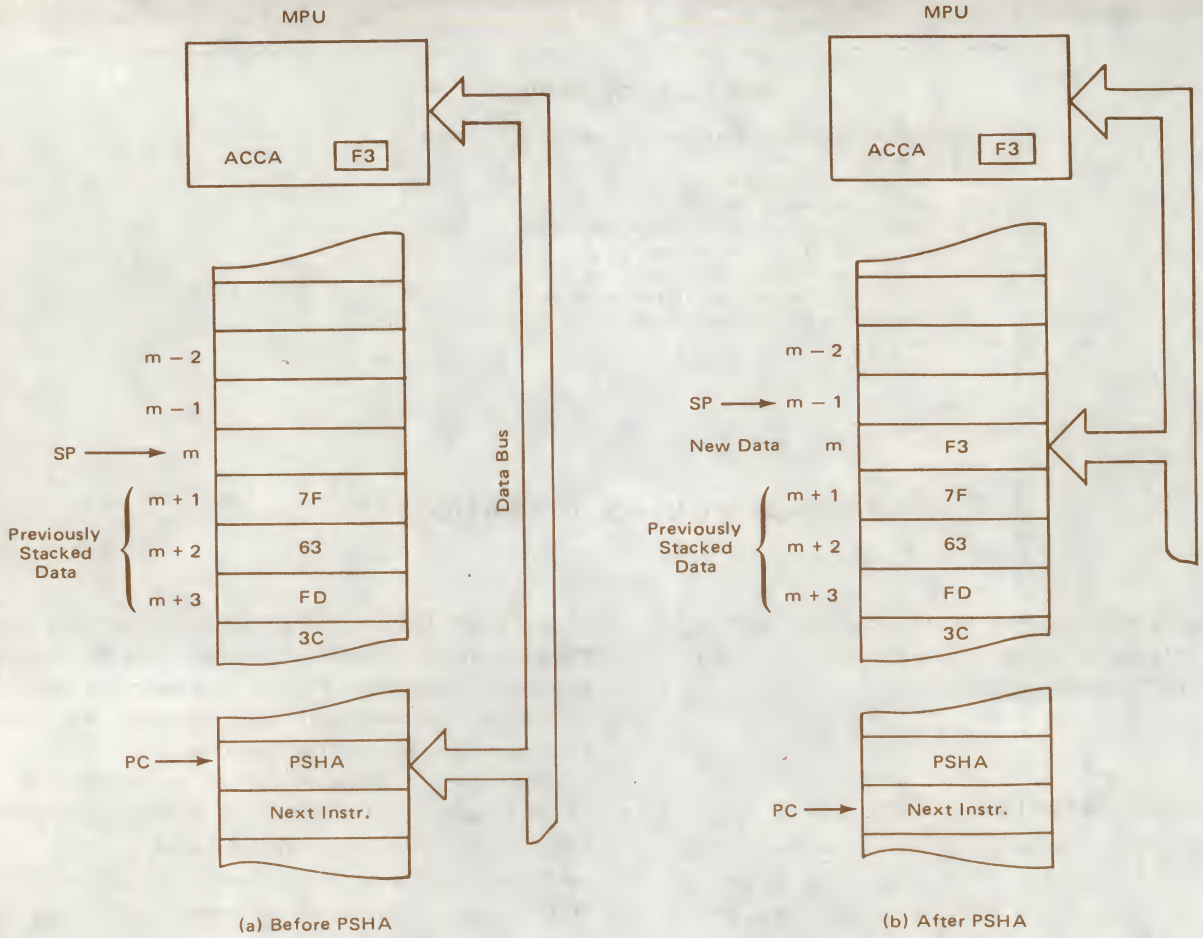
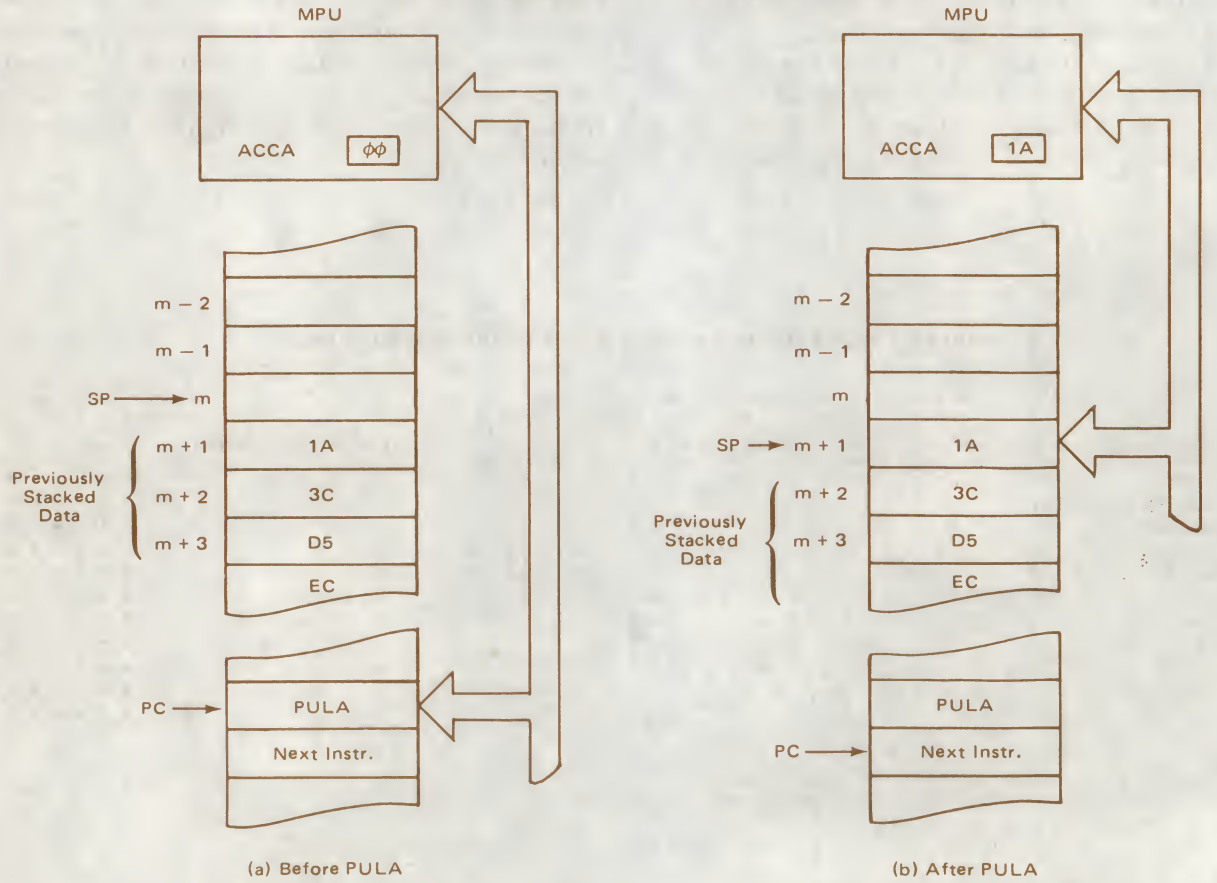


FIGURE 16 – STACK OPERATION, PULL INSTRUCTION



Operation of the Stack Pointer with the Push and Pull instructions is illustrated in Figures 15 and 16. The Push instruction (PSHA) causes the contents of the indicated accumulator (A in this example) to be stored in memory at the location indicated by the Stack Pointer. The Stack Pointer is automatically decremented by one following the storage operation and is "pointing" to the next empty stack location. The Pull instruction (PULA or PULB) causes the last byte stacked to be loaded into the appropriate accumulator. The Stack Pointer is automatically incremented by one just prior to the data transfer so that it will point to the last byte stacked rather than the next empty location. Note that the PULL instruction does not "remove" the data from memory; in the example, 1A is still in location (m + 1) following execution of PULA. A subsequent PUSH instruction would overwrite that location with the new "pushed" data.

Execution of the Branch to Subroutine (BSR) and Jump to Subroutine (JSR) instructions cause a return address to be saved on the stack as shown in Figures 18 through 20. The stack is decremented after each byte of the return address is pushed onto the stack. For both of these instructions, the return address is the memory location following the bytes of code that correspond to the BSR and JSR instruction. The code required for BSR or JSR may be either two or three bytes, depending on whether the JSR is in the indexed (two bytes) or the extended (three bytes) addressing mode. Before it is stacked, the Program Counter is automatically incremented the correct number of times to be pointing at the location of the next instruction. The Return from Subroutine instruction, RTS, causes the return address to be retrieved and loaded into the Program Counter as shown in Figure 21.

There are several operations that cause the status of the MPU to be saved on the stack. The Software Interrupt (SWI) and Wait for Interrupt (WAI) instructions as well as the maskable ($\overline{IRQ}$) and non-maskable ($\overline{NMI}$) hardware interrupts all cause the MPU's internal registers (except for the Stack Pointer itself) to be stacked as shown in Figure 23. MPU status is restored by the Return from Interrupt, RTI, as shown in Figure 22.

Jump and Branch Operation

The Jump and Branch instructions are summarized in Table 9. These instructions are used to control the transfer of operation from one point to another in the control program.

The No Operation instruction, NOP, while included here, is a jump operation in a very limited sense. Its only effect is to increment the Program Counter by one. It is useful during program development as a "stand-in" for some other instruction that is to be determined during debug. It is also used for equalizing the execution time through alternate paths in a control program.

TABLE 9 – JUMP AND BRANCH INSTRUCTIONS

		RELATIVE			INDEX			EXTND			IMPLIED				COND. CODE REG.					
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1					①		
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	②	•	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

① (All) Load Condition Code Register from Stack. (See Special Operations)
② (Bit 1) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.

Execution of the Jump Instruction, JMP, and Branch Always, BRA, affects program flow as shown in Figure 17. When the MPU encounters the Jump (Indexed) instruction, it adds the offset to the value in the Index Register and uses the result as the address of the next instruction to be executed. In the extended addressing mode, the address of the next instruction to be executed is fetched from the two locations immediately following the JMP instruction. The Branch Always (BRA) instruction is similar to the JMP (extended) instruction except that the relative addressing mode applies and the branch is limited to the range within −125 or +127 bytes of the branch instruction itself. The opcode for the BRA instruction requires one less byte than JMP (extended) but takes one more cycle to execute.

The effect on program flow for the Jump to Subroutine (JSR) and Branch to Subroutine (BSR) is shown in Figures 18 through 20. Note that the Program Counter is properly incremented to be pointing at the correct return address before it is stacked. Operation of the Branch to Subroutine and Jump to Subroutine (extended) instruction is similar except for the range. The BSR instruction requires less opcode than JSR (2 bytes versus 3 bytes) and also executes one cycle faster than JSR. The Return from Subroutine, RTS, is used at the end of a subroutine to return to the main program as indicated in Figure 21.

The effect of executing the Software Interrupt, SWI, and the Wait for Interrupt, WAI, and their relationship to the hardware interrupts is shown in Figure 22. SWI causes the MPU contents to be stacked and then fetches the starting address of the interrupt routine from the memory locations that respond to the addresses FFFA and FFFB. Note that as in the case of the subroutine instructions, the Program Counter is incremented to point at the correct return address before being stacked. The Return from Interrupt instruction, RTI, (Figure 22) is used at the end of an interrupt routine to restore control to the main program. The SWI instruction is useful for inserting break points in the control program, that is, it can be used to stop operation and put the MPU registers in memory where they can be examined. The WAI instruction is used to decrease the time required to service a hardware interrupt; it stacks the MPU contents and then waits for the interrupt to occur, effectively removing the stacking time from a hardware interrupt sequence.

FIGURE 17 – PROGRAM FLOW FOR JUMP AND BRANCH INSTRUCTIONS

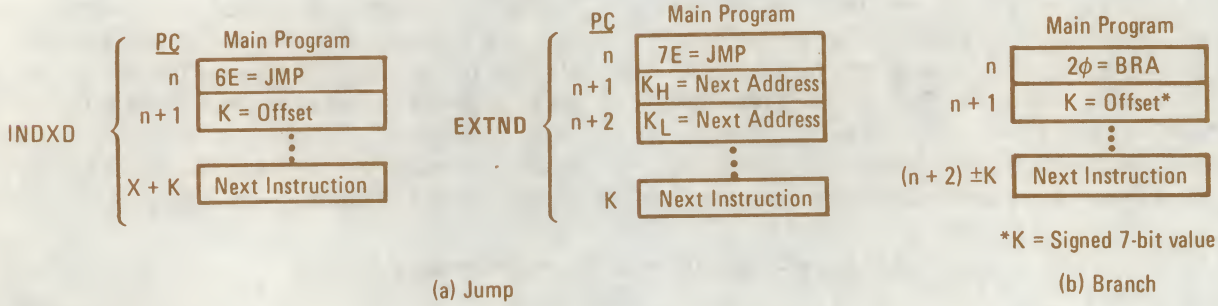


FIGURE 18 – PROGRAM FLOW FOR BSR

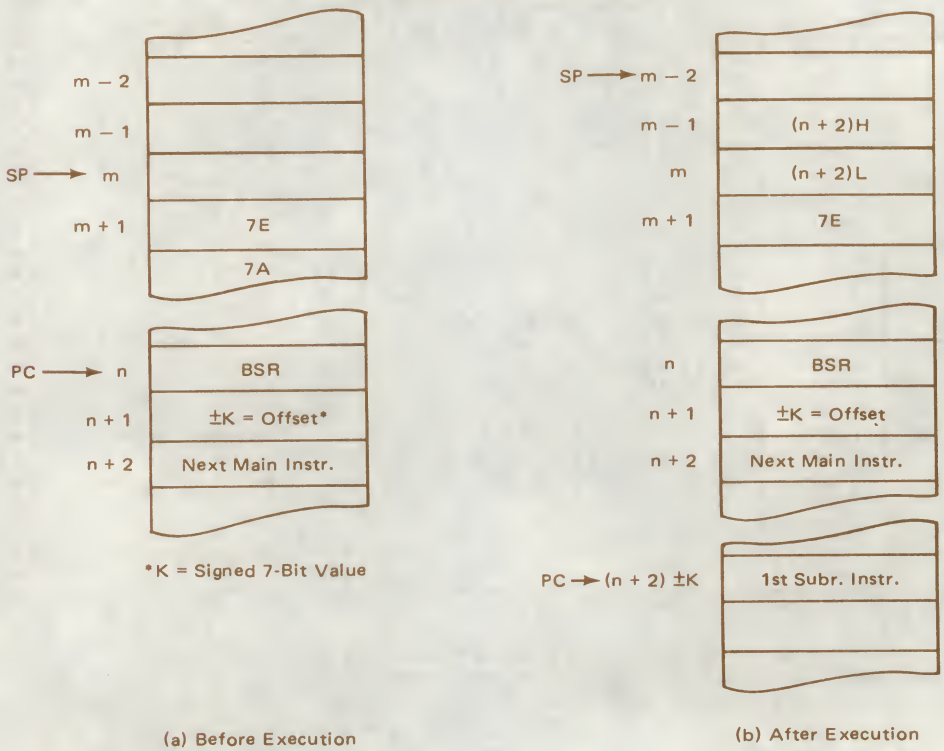


FIGURE 19 – PROGRAM FLOW FOR JSR (EXTENDED)

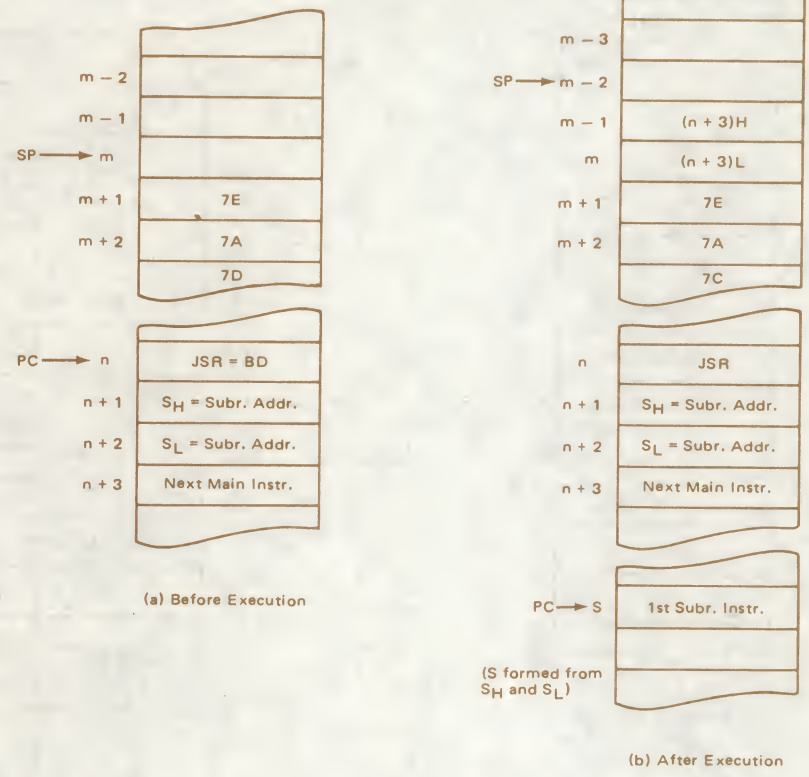


FIGURE 20 – PROGRAM FLOW FOR JSR (INDEXED)

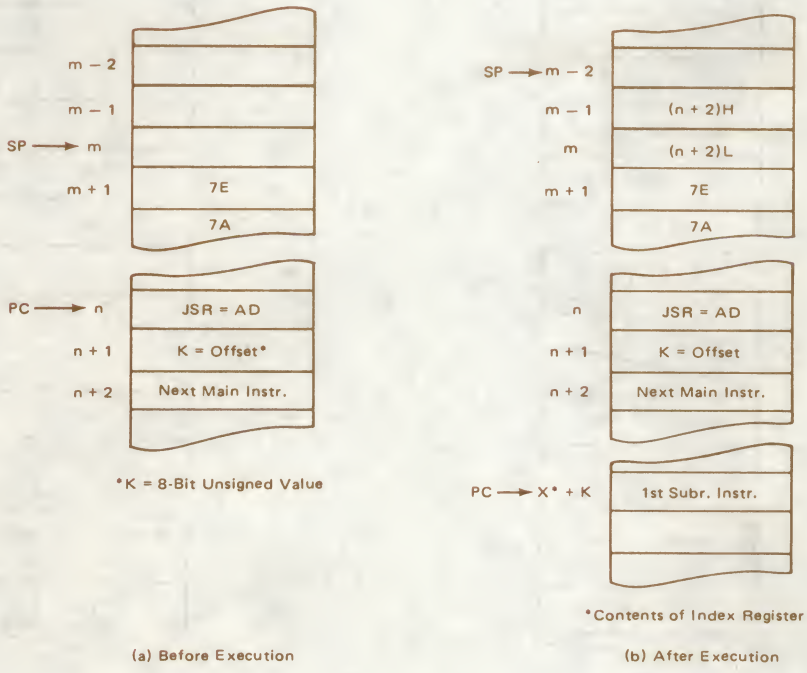
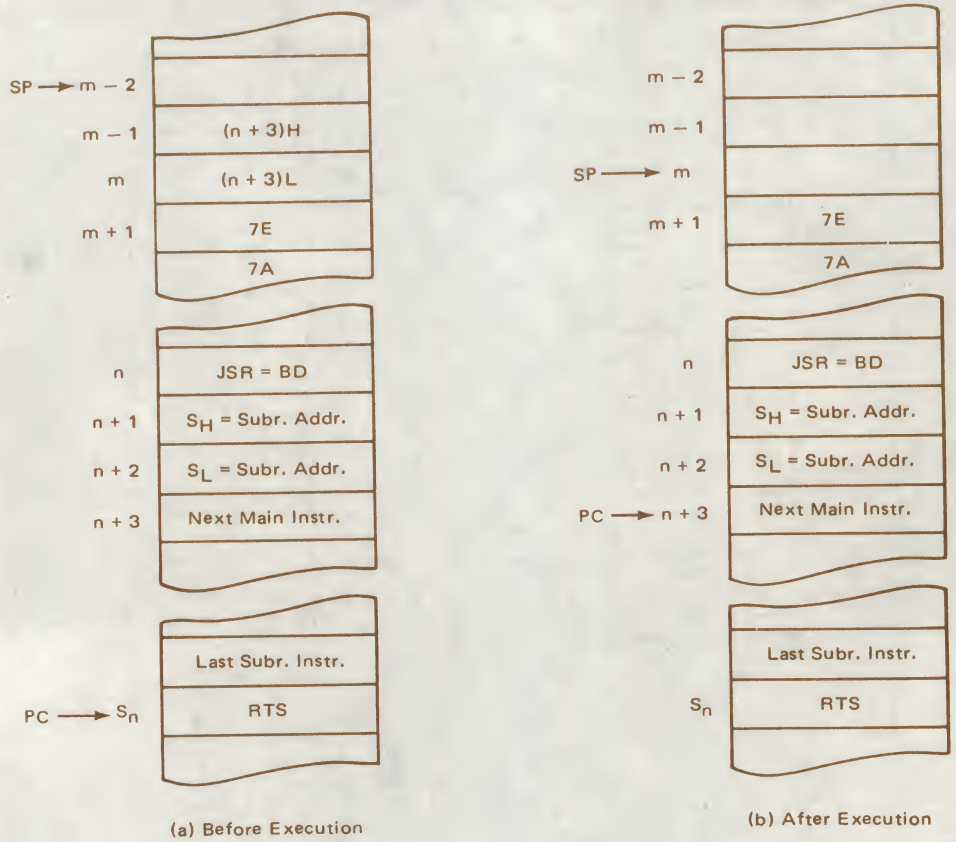


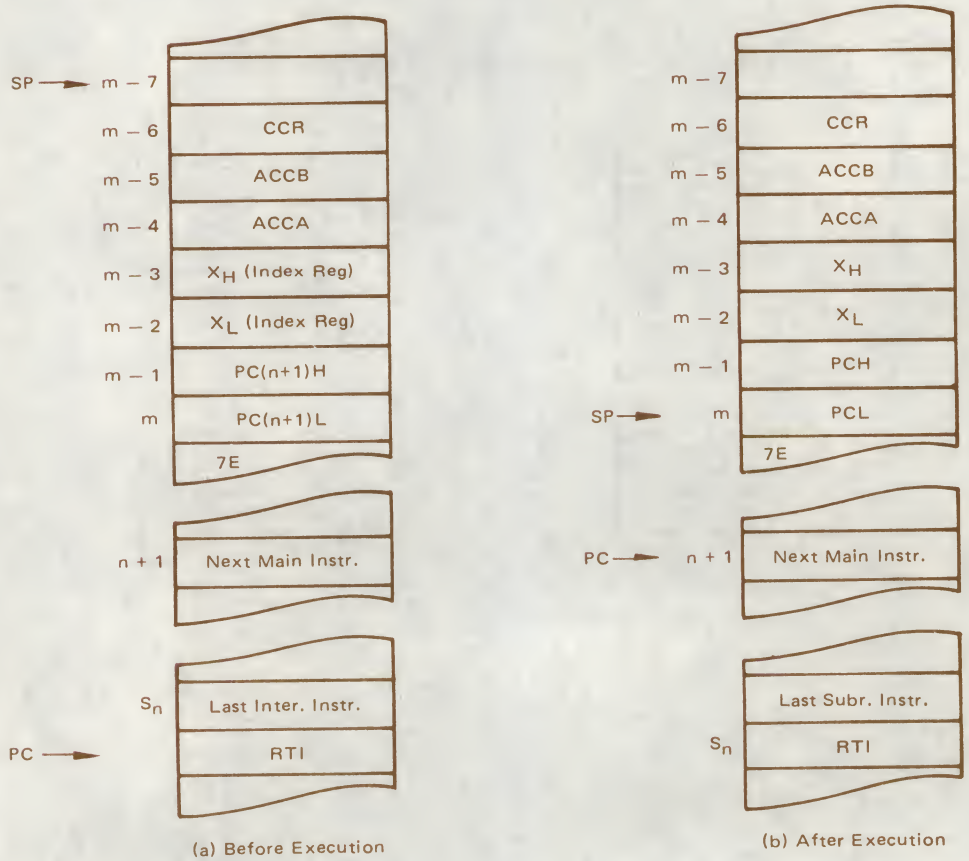
FIGURE 21 – PROGRAM FLOW FOR RTS



(a) Before Execution

(b) After Execution

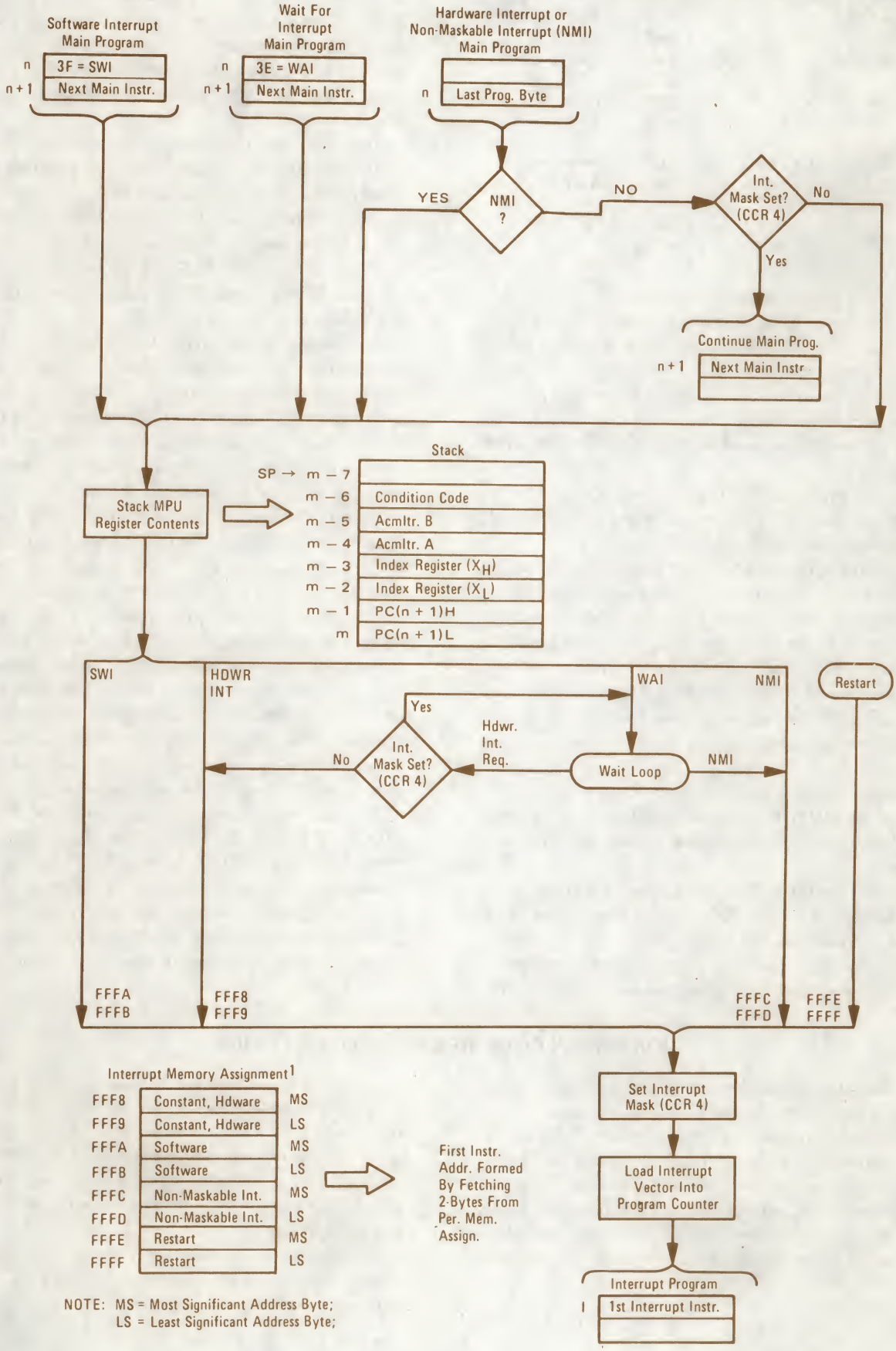
FIGURE 22 – PROGRAM FLOW FOR RTI



(a) Before Execution

(b) After Execution

FIGURE 23 – PROGRAM FLOW FOR INTERRUPTS*



NOTE: MS = Most Significant Address Byte;
LS = Least Significant Address Byte;

FIGURE 24 – CONDITIONAL BRANCH INSTRUCTIONS

BMI :	N = 1 ;	BEQ :	Z = 1 ;
BPL :	N = ϕ ;	BNE :	Z = ϕ ;
BVC :	V = ϕ ;	BCC :	C = ϕ ;
BVS :	V = 1 ;	BCS :	C = 1 ;
BHI :	C + Z = ϕ ;	BLT :	N⊕V = 1 ;
BLS :	C + Z = 1 ;	BGE :	N⊕V = ϕ ;
BLE :	Z + (N⊕V) = 1 ;		
BGT :	Z + (N⊕V) = ϕ ;		

The conditional branch instructions, Figure 24, consists of seven pairs of complementary instructions. They are used to test the results of the preceding operation and either continue with the next instruction in sequence (test fails) or cause a branch to another point in the program (test succeeds).

Four of the pairs are used for simple tests of status bits N, Z, V, and C:

1. Branch on Minus (BMI) and Branch On Plus (BPL) tests the sign bit, N, to determine if the previous result was negative or positive, respectively.

2. Branch On Equal (BEQ) and Branch On Not Equal (BNE) are used to test the zero status bit, Z, to determine whether or not the result of the previous operation was equal to zero. These two instructions are useful following a Compare (CMP) instruction to test for equality between an accumulator and the operand. They are also used following the Bit Test (BIT) to determine whether or not the same bit positions are set in an accumulator and the operand.

3. Branch On Overflow Clear (BVC) and Branch On Overflow Set (BVS) tests the state of the V bit to determine if the previous operation caused an arithmetic overflow.

4. Branch On Carry Clear (BCC) and Branch On Carry Set (BCS) tests the state of the C bit to determine if the previous operation caused a carry to occur. BCC and BCS are useful for testing relative magnitude when the values being tested are regarded as unsigned binary numbers, that is, the values are in the range 00 (lowest) to FF (highest). BCC following a comparison (CMP) will cause a branch if the (unsigned) value in the accumulator is higher than or the same as the value of the operand. Conversely, BCS will cause a branch if the accumulator value is lower than the operand.

The fifth complementary pair, Branch On Higher (BHI) and Branch On Lower or Same (BLS) are in a sense complements to BCC and BCS. BHI tests for both C and Z = 0; if used following a CMP, it will cause a branch if the value in the accumulator is higher than the operand. Conversely, BLS will cause a branch if the unsigned binary value in the accumulator is lower than or the same as the operand.

The remaining two pairs are useful in testing results of operations in which the values are regarded as signed two's complement numbers. This differs from the unsigned binary case in the following sense: In unsigned, the orientation is higher or lower; in signed two's complement, the comparison is between larger or smaller where the range of values is between –128 and +127.

Branch On Less Than Zero (BLT) and Branch On Greater Than Or Equal Zero (BGE) test the status bits for N ⊕ V = 1 and N ⊕ V = 0, respectively. BLT will always cause a branch following an operation in which two negative numbers were added. In addition, it will cause a branch following a CMP in which the value in the accumulator was negative and the operand was positive. BLT will never cause a branch following a CMP in which the accumulator value was positive and the operand negative. BGE, the complement to BLT, will cause a branch following operations in which two positive values were added or in which the result was zero.

The last pair, Branch On Less Than Or Equal Zero (BLE) and Branch On Greater Than Zero (BGT) test the status bits for Z⊕(N + V) = 1 and Z⊕(N + V) = 0, respectively. The action of BLE is identical to that for BLT except that a branch will also occur if the result of the previous result was zero. Conversely, BGT is similar to BGE except that no branch will occur following a zero result.

CONDITION CODE REGISTER OPERATIONS

The Condition Code Register (CCR) is a 6-bit register within the MPU that is useful in controlling program flow during system operation. The bits are defined in Figure 25.

The instructions shown in Table 10 are available to the user for direct manipulation of the CCR. In addition, the MPU automatically sets or clears the appropriate status bits as many of the other instructions on the condition code register was indicated as they were introduced.

A CLI-WAI instruction sequence operated properly with early M6800 processors only if the preceding instruction was odd. (Least Significant Bit = 1.) Similarly it was advisable to precede any SEI instruction with an odd opcode—such as NOP. These precautions are not necessary for M6800 processors indicating manufacture in November, 1977 or later.

Systems which require an interrupt window to be opened under program control should use a CLI-NOP-SEI sequence rather than CLI-SEI.

FIGURE 25 – CONDITION CODE REGISTER BIT DEFINITION

b_5	b_4	b_3	b_2	b_1	b_0
H	I	N	Z	V	C

H = Half-carry; set whenever a carry from b_3 to b_4 of the result is generated by ADD, ABA, ADC; cleared if no b_3 to b_4 carry; not affected by other instructions.

I = Interrupt Mask; set by hardware or software interrupt or SEI instruction; cleared by CLI instruction. (Normally not used in arithmetic operations.) Restored to a zero as a result of an RT1 instruction if I_m stored on the stacked is low.

N = Negative; set if high order bit (b_7) of result is set; cleared otherwise

Z = Zero; set if result = 0; cleared otherwise.

V = Overlow; set if there was arithmetic overflow as a result of the operation; cleared otherwise.

C = Carry; set if there was a carry from the most significant bit (b_7) of the result; cleared otherwise.

TABLE 10 – CONDITION CODE REGISTER INSTRUCTIONS

		IMPLIED				COND. CODE REG.					
						5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	BOOLEAN OPERATION	H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0→C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0→V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1→C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1→I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1→V	•	•	•	•	S	•
Acmltr A→CCR	TAP	06	2	1	A→CCR	———	———	———	①	———	———
CCR→Acmltr A	TPA	07	2	1	CCR→A	•	•	•	•	•	•

R = Reset
S = Set
• = Not affected
① (ALL) Set according to the contents of Accumulator A.

ADDRESSING MODES

The MPU operates on 8-bit binary numbers presented to it via the Data Bus. A given number (byte) may represent either data or an instruction to be executed, depending on where it is encountered in the control program. The M6800 has 72 unique instructions, however, it recognizes and takes action on 197 of the 256 possibilities that can occur using an 8-bit word length. This larger number of instructions results from the fact that many of the executive instructions have more than one addressing mode.

These addressing modes refer to the manner in which the program causes the MPU to obtain its instructions and data. The programmer must have a method for addressing the MPU's internal registers and all of the external memory locations.

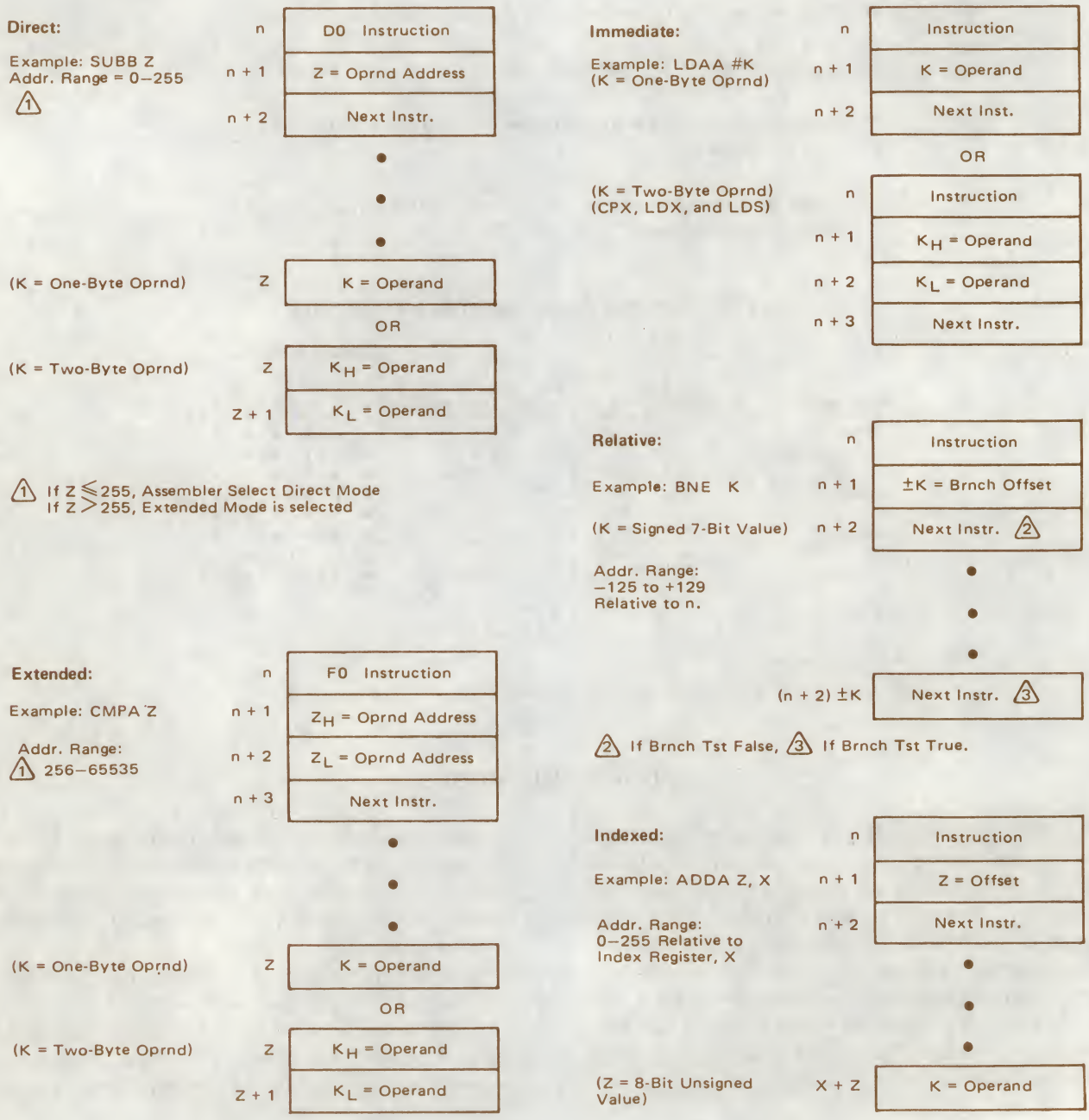
Selection of the desired addressing mode is made by the user as the source statements are written. Translation into appropriate opcode then depends on the method used. If manual translation is used, the addressing mode is inherent in the opcode. For example, the Immediate,

Direct, Indexed, and Extended modes may all be used with the ADD instruction. The proper mode is determined by selecting (hexidecimal notation) 8B, 9B, AB, or BB, respectively.

The source statement format includes adequate information for the selection if an assembler program is used to generate the opcode. For instance, the Immediate mode is selected by the Assembler whenever it encounters the "#" symbol in the operand field. Similarly, an "X" in the operand field causes the Indexed mode to be selected. Only the Relative mode applies to the branch instructions, therefore, the mnemonic instruction itself is enough for the Assembler to determine addressing mode.

For the instructions that use both Direct and Extended modes, the Assembler selects the Direct mode if the operand value is in the range 0-255 and Extended otherwise. There are a number of instructions for which the Extended mode is valid but the Direct is not. For these instructions, the Assembler automatically selects the Extended mode even if the operand is in the 0-255 range. The addressing modes are summarized in Figure 26.

FIGURE 26 – ADDRESSING MODE SUMMARY



Inherent (Includes "Accumulator Addressing" Mode)

The successive fields in a statement are normally separated by one or more spaces. An exception to this rule occurs for instructions that use dual addressing in the operand field and for instructions that must distinguish between the two accumulators. In these cases, A and B are "operands" but the space between them and the operator may be omitted. This is commonly done, resulting in apparent four character mnemonics for those instructions.

The addition instruction, ADD, provides an example of dual addressing in the operand field:

	Operator	Operand	Comment
	ADDA	MEM12	ADD CONTENTS OF MEM12 TO ACCA
or	ADDB	MEM12	ADD CONTENTS OF MEM12 TO ACCB

The example used earlier for the test instruction, TST, also applies to the accumulators and uses the "accumulator addressing mode" to designate which of the two accumulators is being tested:

	Operator	Comment
	TSTB	TEST CONTENTS OF ACCB
or	TSTA	TEST CONTENTS OF ACCA

A number of the instructions either alone or together with an accumulator operand contain all of the address information that is required, that is, "inherent" in the instruction itself. For instance, the instruction ABA causes the MPU to add the contents of accumulators A and B together and place the result in accumulator A. The instruction INCB, another example of "accumulator addressing", causes the contents of accumulator B to be increased by one. Similarly, INX, increment the Index Register, causes the contents of the Index Register to be increased by one.

Program flow for instructions fo this type is illustrated in Figures 27 and 28. In these figures, the general case is shown on the left and a specific example is shown on the right. Numerical examples are in decimal notation. Instructions of this type require only one byte of opcode. Cycle-by-cycle operation of the inehrent mode is shown in Table 11.

FIGURE 27 – INHERENT ADDRESSING

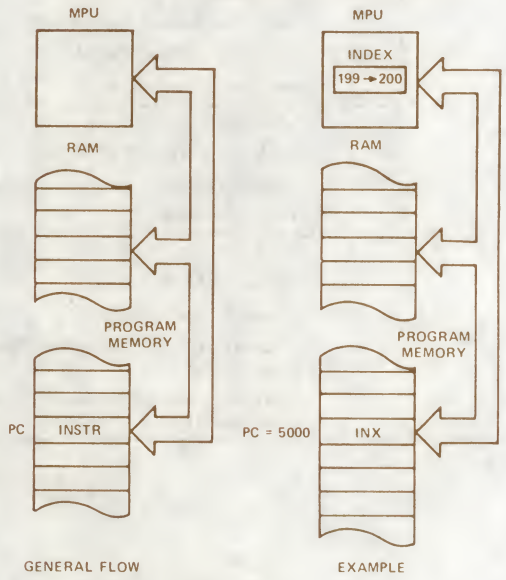


FIGURE 28 – ACCUMULATOR ADDRESSING

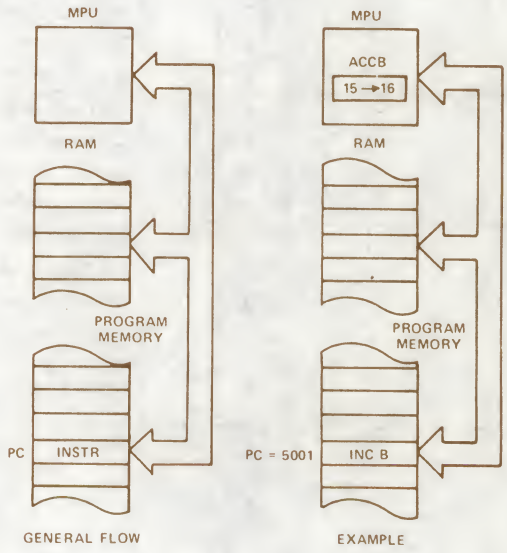


TABLE 11 – INHERENT MODE CYCLE BY CYCLE OPERATION

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
DES DEX INS INX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Previous Register Contents	1	Irrelevant Data (Note 1)
		4	0	New Register Contents	1	Irrelevant Data (Note 1)
PSH	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Accumulator Data
		4	0	Stack Pointer – 1	1	Accumulator Data
PUL	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Operand Data from Stack
TSX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	0	New Index Register	1	Irrelevant Data (Note 1)
TXS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Index Register	1	Irrelevant Data
		4	0	New Stack Pointer	1	Irrelevant Data
RTS	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
		5	1	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)

TABLE 11 – INHERENT MODE CYCLE BY CYCLE OPERATION (Continued)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
WAI	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6 (Note 3)	1	Contents of Cond. Code Register
RTI	10	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Contents of Cond. Code Register from Stack
		5	1	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	1	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	1	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	1	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	1	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	1	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 1)
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	0	Stack Pointer – 7	1	Irrelevant Data (Note 1)
		11	1	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	1	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

Note 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

Note 2. Data is ignored by the MPU.

Note 3. While the MPU is waiting for the interrupt, Bus Available will go high indicating the following states of the control lines: VMA is low; Address Bus, R/$\overline{W}$, and Data Bus are all in the high impedance state.

Immediate Addressing Mode – In the Immediate addressing mode, the operand is the value that is to be operated on. For instance, the instruction

Operator	Operand	Comment
LDAA	#25	LOAD 25 INTO ACCA

causes the MPU to "immediately load accumulator A with the value 25"; no further address reference is required. The Immediate mode is selected by preceding the operand value with the "#" symbol. Program flow for this addressing mode is illustrated in Figure 29.

The operand format allows either properly defined symbols or numerical values. Except for the instructions CPX, LDX, and LDS, the operand may be any value in the range 0 to 255. Since Compare Index Register (CPX), Load Index Register (LDX), and Load Stack Pointer (LDS), require 16-bit values, the immediate mode for these three instructions require two-byte operands. In the Immediate addressing mode, the "address" of the operand is effectively the memory location immediately following the instruction itself. Table 12 shows the cycle-by-cycle operation for the immediate addressing mode.

Direct and Extended Addressing Modes – In the Direct and Extended modes of addressing, the operand field of the source statement is the *address* of the value that is to be operated on. The Direct and Extended modes differ only in the range of memory locations to which they can direct the MPU. Direct addressing generates a single 8-bit operand and, hence, can address only memory locations 0 through 255; a two byte operand is generated for Extended addressing, enabling the MPU to reach the remaining memory locations, 256 through 65535. An example of Direct addressing and its effect on program flow is illustrated in Figure 30.

FIGURE 29 – IMMEDIATE ADDRESSING MODE

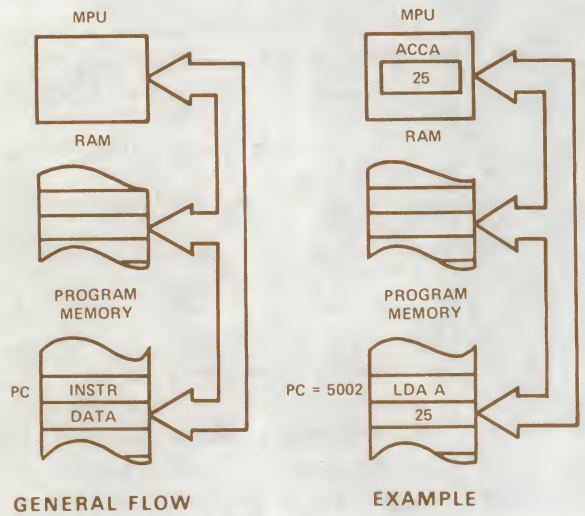


FIGURE 30 – DIRECT ADDRESSING MODE

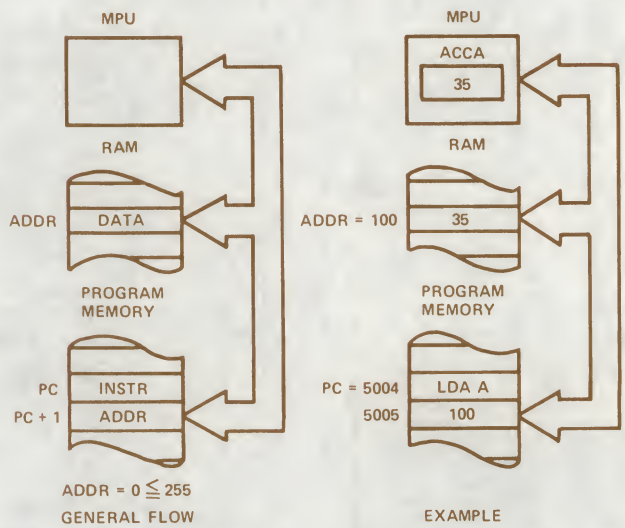


TABLE 12 – IMMEDIATE MODE CYCLE BY CYCLE OPERATION

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data
CPX LDS LDX	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	1	Op Code Address + 2	1	Operand Data (Low Order Byte)

The MPU, after encountering the opcode for the instruction LDAA (Direct) at memory location 5004 (Program Counter = 5004), looks in the next location, 5005, for the address of the operand. It then sets the program counter equal to the value found there (100 in the example) and fetches the operand, in this case a value to be loaded into accumulator A, from that location. For instructions requiring a two-byte operand such as LDX (load the Index Register), the operand bytes would be retrieved from locations 100 and 101. Table 13 shows the cycle-by-cycle operation for the direct mode of addressing.

Extended addressing, Figure 31, is similar except that a two-byte address is obtained from locations 5007 and 5008 after the LDAB (Extended) opcode shows up in location 5006. Extended addressing can be thought of as the "standard" addressing mode, that is, it is a method of reaching anyplace in memory. Direct addressing, since only one address byte is required, provides a faster method of processing data and generates fewer bytes of control code. In most applications, the direct addressing range, memory locations 0-255, are reserved for RAM. They are used for data buffering and temporary storage of system variables, the area in which faster addressing is of most value. Cycle-by-cycle operation is shown in Table 14 for Extended Addressing.

TABLE 13 – DIRECT MODE CYCLE BY CYCLE OPERATION

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data
CPX LDS LDX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data (High Order Byte)
		4	1	Operand Address + 1	1	Operand Data (Low Order Byte)
STA	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address
		3	0	Destination Address	1	Irrelevant Data (Note 1)
		4	1	Destination Address	0	Data from Accumulator
STS STX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	0	Address of Operand	1	Irrelevant Data (Note 1)
		4	1	Address of Operand	0	Register Data (High Order Byte)
		5	1	Address of Operand + 1	0	Register Data (Low Order Byte)

Note 1. If device which is address during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

FIGURE 31 – EXTENDED ADDRESSING MODE

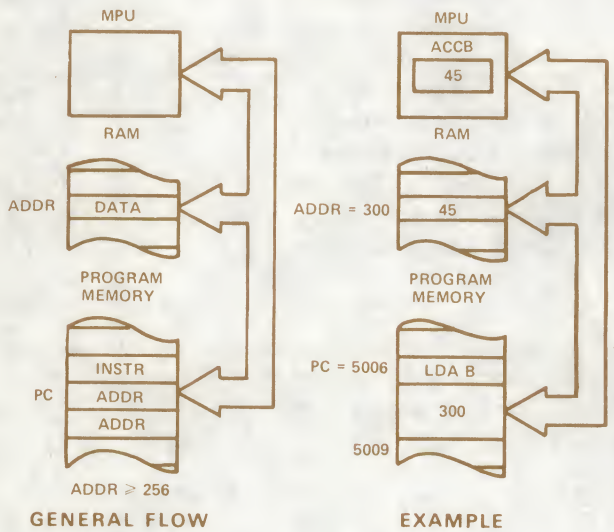


TABLE 14 – EXTENDED MODE CYCLE BY CYCLE

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
STS STX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	0	Address of Operand	1	Irrelevant Data (Note 1)
		5	1	Address of Operand	0	Operand Data (High Order Byte)
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)
JSR	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction
		5	1	Stack Pointer	0	Return Address (Low Order Byte)
		6	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		7	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		8	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (Note 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (Note 1)
		6	1/0 (Note 2)	Address of Operand	0	New Operand Data (Note 2)

Note 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

Note 2. For TST, VMA = 0 and Operand data does not change.

Relative Address Mode – In both the Direct and Extended modes, the address obtained by the MPU is an absolute numerical address. The Relative addressing mode, implemented for the MPU's branch instructions, specifies a memory location relative to the Program Counter's current location. Branch instructions generate two bytes of machine code, one for the instruction opcode and one for the "relative" address (see Figure 32). Since it is desirable to be able to branch in either direction, the 8-bit address byte is interpreted as a signed 7-bit value; the 8th bit of the operand is treated as a sign bit, "0" = plus and "1" = minus. The remaining seven bits represent the numerical value. This results in a relative addressing range of ±127 with respect to the location of the branch instruction itself. However, the branch range is computed with respect to the next instruction that would be executed if the branch conditions are not satisfied. Since two bytes are generated, the next instruction is located at PC + 2. If D is defined as the address of the branch destination, the range is then:

$$(PC + 2) - 127 \leqslant D \leqslant (PC + 2) + 127$$

or

$$PC - 125 \leqslant D \leqslant PC + 129$$

that is, the destination of the branch instruction must be within −125 to +129 memory locations of the branch instruction itself. For transferring control beyond this range, the unconditional jump (JMP), jump to subroutine (JSR), and return from subroutine (RTS) are used.

In Figure 32, when the MPU encounters the opcode for BEQ (Branch if result of last instruction was zero), it tests the Zero bit in the Condition Code Register. If that bit is "0", indicating a non-zero result, the MPU continues execution with the next instruction (in location 5010 in Figure 32). If the previous result was zero, the branch condition is satisfied and the MPU adds the offset, 15 in this case, to PC + 2 and branches to location 5025 for the next instruction.

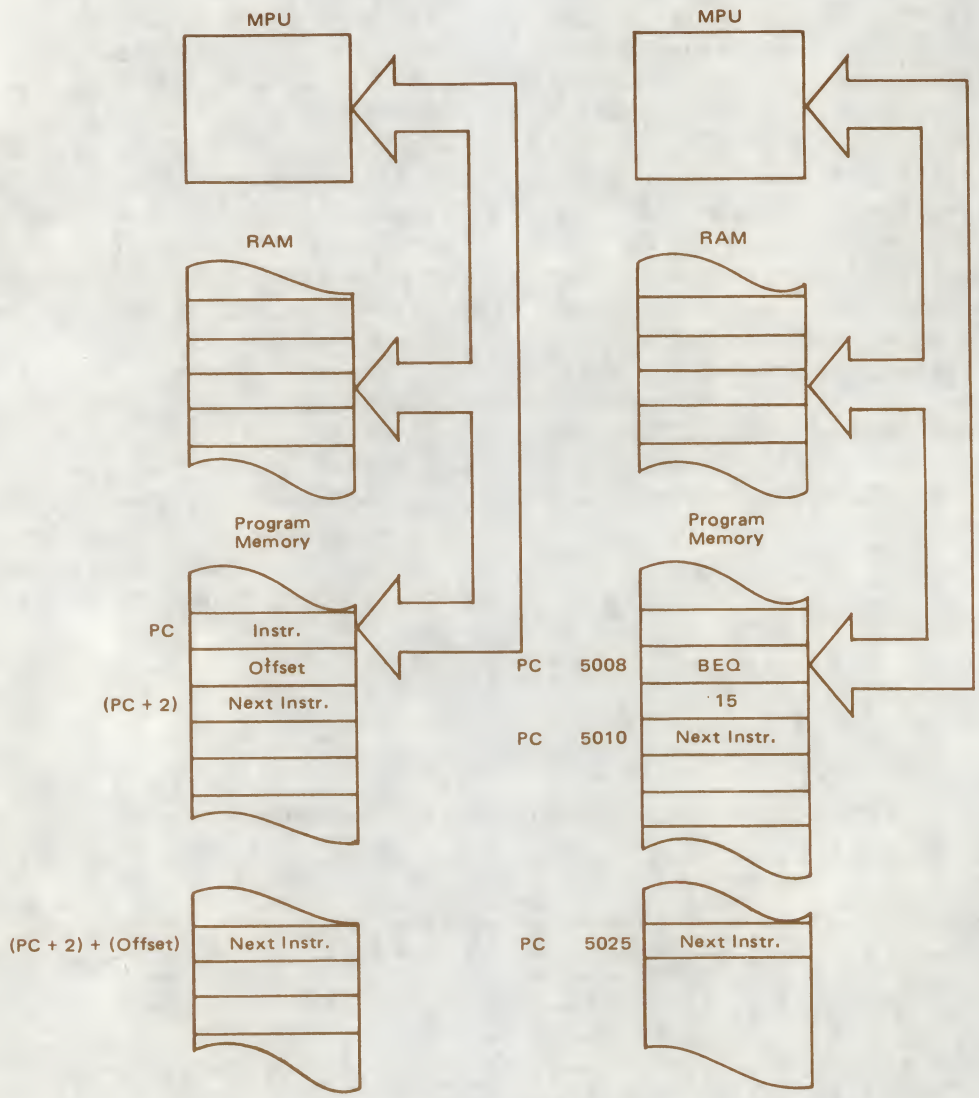
The branch instructions allow the programmer to efficiently direct the MPU to one point or another in the control program depending on the outcome of test results. Since the control program is normally in read-only memory and cannot be changed, the relative address used in execution of branch instructions is a constant numerical value. Cycle-by-cycle operation is shown in Table 15 for relative addressing.

TABLE 15 – RELATIVE MODE CYCLE-BY-CYCLE OPERATION

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		4	0	Branch Address	1	Irrelevant Data (Note 1)
BSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		8	0	Subroutine Address	1	Irrelevant Data (Note 1)

Note 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

FIGURE 32 – RELATIVE ADDRESSING MODE



Indexed Addressing Mode – With Indexed addressing, the numerical address is variable and depend on the current contents of the Index Register. A source statement such as

Operator	Operand	Comment
STAA	X	PUT A IN INDEXED LOCATION

causes the MPU to store the contents of accumulator A in the memory location specified by the contents of the Index Register (recall that the label "X" is reserved to designate the Index Register). Since there are instructions for manipulating X during program execution (LDX, INX, DEX, etc.), the Indexed addressing mode provides a dynamic "on the fly" way to modify program activity.

The operand field can also contain a numerical value that will be automatically added to X during execution. This format is illustrated in Figure 33.

When the MPU encounters the LDAB (Indexed) opcode in location 5006, it looks in the next memory location for the value to be added to X (5 in the example) and calculates the required address by adding 5 to the present Index Register value of 400. In the operand format, the offset may be represented by a label or a numerical value in the range 0-255 as in the example. In the earlier example, STAA X, the operand is equivalent to 0,X, that is, the 0 may be omitted when the desired address is equal to X. Table 16 shows the cycle-by-cycle operation for the Indexed Mode of Addressing.

FIGURE 33 – INDEXED ADDRESSING MODE

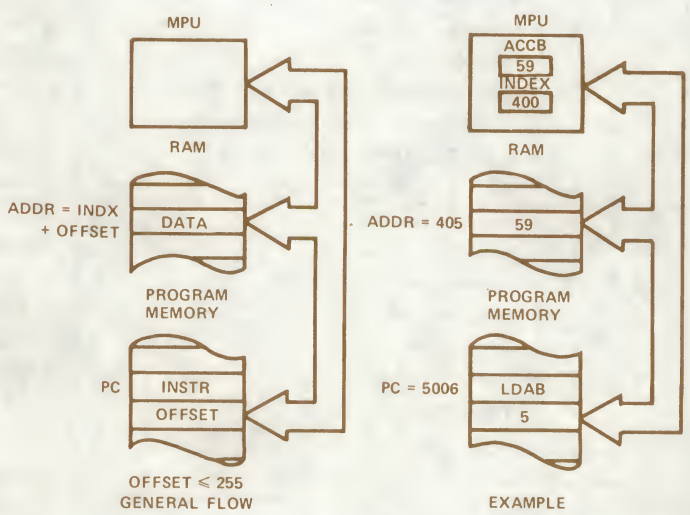


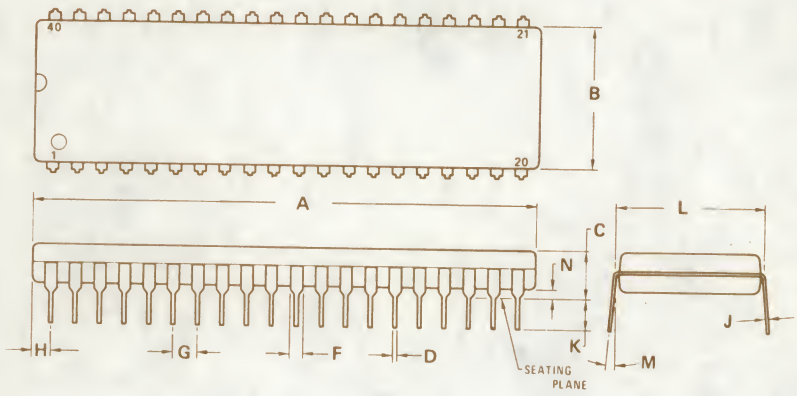
TABLE 16 – INDEXED MODE CYCLE BY CYCLE

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INDEXED						
JMP	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data
CPX LDS LDX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data (High Order Byte)
		6	1	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STA	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Current Operand Data
		6	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		7	1/0 (Note 2)	Index Register Plus Offset	0	New Operand Data (Note 2)
STS STX	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
		7	1	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
JSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Index Register	1	Irrelevant Data (Note 1)
		8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)

Note 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

Note 2. For TST, VMA = 0 and Operand data does not change.

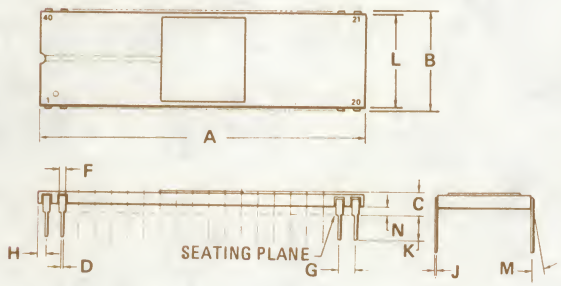
PACKAGE DIMENSIONS



NOTES:
1. LEADS TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION (DIM "D").
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	51.82	52.32	2.040	2.060
B	13.72	14.22	0.540	0.560
C	4.57	5.08	0.180	0.200
D	0.36	0.51	0.014	0.020
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.30	0.008	0.012
K	3.05	3.56	0.120	0.140
L	15.24 BSC		0.600 BSC	
M	0°	10°	0°	10°
N	0.51	1.02	0.020	0.040

CASE 711-02
(PLASTIC)



	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	50.29	51.31	1.980	2.020
B	14.86	15.62	0.585	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.60	15.37	0.575	0.605
M	0°	10°	0°	10°
N	0.51	1.52	0.020	0.060

NOTE:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX. MAT'L CONDITION.

CASE 715-02
(CERAMIC)

MOTOROLA Semiconductor Products Inc.

133-E78/10.0 Printed in Switzerland